

# Yandex.Maps JavaScript API Reference

Version 2.1.72

10.12.2018



Yandex.Maps JavaScript API Reference. Version 2.1.72. Version

Document build date: 10.12.2018

This volume is a part of Yandex technical documentation.

© 2008—2018 Yandex LLC. All rights reserved.

## Copyright Disclaimer

Yandex (and its applicable licensor) has exclusive rights for all results of intellectual activity and equated to them means of individualization, used for development, support, and usage of the service Yandex.Maps JavaScript API Reference. It may include, but not limited to, computer programs (software), databases, images, texts, other works and inventions, utility models, trademarks, service marks, and commercial denominations. The copyright is protected under provision of Part 4 of the Russian Civil Code and international laws.

You may use Yandex.Maps JavaScript API Reference or its components only within credentials granted by the Terms of Use of Yandex.Maps JavaScript API Reference or within an appropriate Agreement.

Any infringements of exclusive rights of the copyright owner are punishable under civil, administrative or criminal Russian laws.

## Contact information

Yandex LLC

<https://www.yandex.com>

Ten.: +7 495 739 7000

Email: [pr@yandex-team.ru](mailto:pr@yandex-team.ru)

16 L'va Tolstogo St., Moscow, Russia 119021

# Contents

Yandex.Maps API Reference.....	11
Balloon.....	11
Events details.....	15
behavior.....	15
behavior.DbClickZoom.....	15
behavior.Drag.....	17
behavior.LeftMouseButtonMagnifier.....	19
behavior.MultiTouch.....	21
behavior.RightMouseButtonMagnifier.....	22
behavior.RouteEditor.....	24
behavior.Ruler.....	26
behavior.ScrollZoom.....	29
behavior.storage.....	31
borders.....	31
borders.load.....	31
Circle.....	33
Fields details.....	42
clusterer.....	43
clusterer.addon.....	43
clusterer.Balloon.....	44
clusterer.Hint.....	47
Clusterer.....	49
Fields details.....	54
Events details.....	54
Methods details.....	54
ClusterPlacemark.....	57
Fields details.....	67
Methods details.....	68
collection.....	68
collection.Item.....	68
Collection.....	71
Events details.....	73
Methods details.....	73
control.....	76
control.Button.....	76
control.FullscreenControl.....	82
control.GeolocationControl.....	88
control.ListBox.....	92
control.ListBoxItem.....	100
control.Manager.....	104
control.RouteButton.....	110
control.RouteEditor.....	114
control.RoutePanel.....	119
control.RulerControl.....	121
control.SearchControl.....	125
control.storage.....	136
control.TrafficControl.....	137
control.TypeSelector.....	143
control.ZoomControl.....	150
coordSystem.....	152
coordSystem.cartesian.....	152
coordSystem.geo.....	153

---

data.....	154
data.Manager.....	154
DomEvent.....	158
Methods details.....	160
domEvent.....	160
domEvent.manager.....	160
domEvent.MultiPointer.....	163
domEvent.MultiTouch.....	164
domEvent.Pointer.....	165
domEvent.Touch.....	166
event.....	167
event.Group.....	167
event.Manager.....	168
event.Mapper.....	171
Event.....	172
Methods details.....	174
formatter.....	175
Methods details.....	176
geocode.....	176
GeocodeResult.....	179
Methods details.....	182
geolocation.....	183
Methods details.....	184
geometry.....	185
geometry.base.....	185
geometry.Circle.....	202
geometry.json.....	205
geometry.LineString.....	209
geometry.pixel.....	214
geometry.Point.....	228
geometry.Polygon.....	230
geometry.Rectangle.....	236
geometryEditor.....	239
geometryEditor.Circle.....	239
geometryEditor.LineString.....	241
geometryEditor.model.....	253
geometryEditor.Point.....	265
geometryEditor.Polygon.....	267
geometryEditor.view.....	279
geoObject.....	282
geoObject.addon.....	282
geoObject.Balloon.....	284
geoObject.Hint.....	287
geoObject.Sequence.....	290
GeoObject.....	295
Fields details.....	308
Events details.....	309
GeoObjectCollection.....	310
Events details.....	316
Methods details.....	316
geoQuery.....	318
GeoQueryResult.....	320
Methods details.....	327
geoXml.....	355
geoXml.load.....	355
getZoomRange.....	356
graphics.....	357
graphics.style.....	357

---

Hint.....	359
Hotspot.....	362
hotspot.....	365
hotspot.layer.....	365
hotspot.Layer.....	374
hotspot.ObjectSource.....	385
interactivityModel.....	390
interactivityModel.storage.....	390
Interfaces.....	390
IBalloon.....	390
IBalloonLayout.....	392
IBalloonManager.....	395
IBalloonOwner.....	398
IBaseCircleGeometry.....	398
IBaseGeometry.....	400
IBaseLinearRingGeometry.....	401
IBaseLineStringGeometry.....	404
IBasePointGeometry.....	406
IBasePolygonGeometry.....	407
IBaseRectangleGeometry.....	410
IBehavior.....	411
ICanvasTile.....	413
IChild.....	415
IChildOnMap.....	416
ICircleGeometry.....	417
ICircleGeometryAccess.....	419
ICollection.....	422
IContainerPane.....	424
IControl.....	426
IControlParent.....	428
ICoordSystem.....	429
ICopyrightsAccessor.....	433
ICopyrightsProvider.....	433
ICustomizable.....	435
IDataManager.....	436
IDomEvent.....	437
IDomEventEmitter.....	439
IDomTile.....	442
IEvent.....	444
IEventController.....	446
IEventEmitter.....	447
IEventGroup.....	448
IEventManager.....	449
IEventPane.....	453
IEventTrigger.....	455
IEventWorkflowController.....	456
IExpandableControlLayout.....	457
IFreezable.....	460
IGeocodeProvider.....	461
IGeometry.....	464
IGeometryEditor.....	466
IGeometryEditorChildModel.....	468
IGeometryEditorModel.....	469
IGeometryEditorRootModel.....	470
IGeometryJson.....	471
IGeoObject.....	471
IGeoObjectCollection.....	475
IGeoObjectPopupData.....	481

---

IGeoObjectSequence.....	481
IGroupControlLayout.....	484
IHint.....	487
IHintManager.....	489
IHintOwner.....	491
IHotspot.....	491
IHotspotLayerObject.....	494
IHotspotObjectSource.....	498
IHotspotShape.....	499
Iterator.....	503
ILayer.....	504
ILayout.....	507
ILinearRingGeometryAccess.....	511
ILineStringGeometry.....	516
ILineStringGeometryAccess.....	519
IMapAction.....	524
IMapObjectCollection.....	525
IMapState.....	526
IMultiRouteModelJson.....	527
IMultiRouteParams.....	528
IMultiRouteReferencePoint.....	530
IMultiRouterRouteBalloon.....	530
IOptionManager.....	531
IOverlay.....	535
IPane.....	539
IPanorama.....	541
IPanoramaConnection.....	544
IPanoramaConnectionArrow.....	544
IPanoramaConnectionMarker.....	545
IPanoramaGraph.....	546
IPanoramaGraphEdge.....	547
IPanoramaGraphNode.....	547
IPanoramaMarker.....	548
IPanoramaMarkerIcon.....	549
IPanoramaMarkerIconSet.....	550
IPanoramaTileLevel.....	550
IParentOnMap.....	551
IPixelCircleGeometry.....	552
IPixelGeometry.....	554
IPixelLineStringGeometry.....	556
IPixelMultiLineGeometry.....	558
IPixelMultiPolygonGeometry.....	559
IPixelPointGeometry.....	563
IPixelPolygonGeometry.....	564
IPixelRectangleGeometry.....	567
IPointGeometry.....	569
IPointGeometryAccess.....	571
IPolygonGeometry.....	572
IPolygonGeometryAccess.....	575
IPopup.....	581
IPopupManager.....	584
IPositioningContext.....	588
IProjection.....	589
IPromiseProvider.....	590
IRatioMap.....	591
IRectangleGeometry.....	591
IRectangleGeometryAccess.....	593
IRoutePanel.....	595

---

ISearchControlLayout.....	598
ISearchProvider.....	602
ISelectableControl.....	604
ISelectableControlLayout.....	606
IShape.....	609
ISuggestProvider.....	611
ISuggestViewLayout.....	613
ITile.....	613
ITrafficControlLayout.....	615
ITrafficProvider.....	618
ITransportProperties.....	619
IZoomControlLayout.....	620
layer.....	623
layer.storage.....	623
layer.tile.....	623
layer.tileContainer.....	626
Layer.....	630
Methods details.....	635
LayerCollection.....	637
Methods details.....	640
layout.....	642
layout.Image.....	642
layout.ImageWithContent.....	646
layout.PieChart.....	649
layout.storage.....	654
layout.templateBased.....	655
LoadingObjectManager.....	659
Fields details.....	666
Methods details.....	666
Map.....	668
Fields details.....	676
Events details.....	679
Methods details.....	681
map.....	690
map.action.....	690
map.addon.....	699
map.Balloon.....	700
map.behavior.....	702
map.Container.....	706
map.Converter.....	711
map.Copyrights.....	712
map.GeoObjects.....	715
map.Hint.....	721
map.layer.....	723
map.margin.....	726
map.pane.....	731
map.ZoomRange.....	735
MapEvent.....	736
MapType.....	738
Methods details.....	739
mapType.....	740
mapType.storage.....	740
meta.....	740
Fields details.....	741
modules.....	742
modules.define.....	742
modules.isDefined.....	744
modules.require.....	745

---

Monitor.....	746
Methods details.....	747
multiRouter.....	749
multiRouter.bicycle.....	749
multiRouter.driving.....	761
multiRouter.Editor.....	780
multiRouter.EditorAddon.....	789
multiRouter.masstransit.....	797
multiRouter.MultiRoute.....	830
multiRouter.MultiRouteModel.....	839
multiRouter.pedestrian.....	844
multiRouter.ViaPoint.....	862
multiRouter.ViaPointModel.....	866
multiRouter.WayPoint.....	868
multiRouter.WayPointModel.....	872
ObjectManager.....	875
Fields details.....	880
Methods details.....	880
objectManager.....	886
objectManager.addon.....	886
objectManager.Balloon.....	888
objectManager.ClusterCollection.....	892
objectManager.Hint.....	898
objectManager.ObjectCollection.....	902
objectManager.OverlayCollection.....	908
option.....	918
option.Manager.....	918
option.presetStorage.....	923
overlay.....	933
overlay.Circle.....	933
overlay.hotspot.....	937
overlay.html.....	959
overlay.Pin.....	979
overlay.Placemark.....	984
overlay.Polygon.....	989
overlay.Polyline.....	994
overlay.Rectangle.....	998
overlay.storage.....	1002
pane.....	1003
pane.EventsPane.....	1003
pane.MovablePane.....	1006
pane.StaticPane.....	1009
panorama.....	1011
panorama.Base.....	1011
panorama.createPlayer.....	1017
panorama.isSupported.....	1018
panorama.locate.....	1018
panorama.Manager.....	1019
panorama.Player.....	1022
Panorama.....	1030
Methods details.....	1032
Placemark.....	1033
Fields details.....	1043
Polygon.....	1043
Fields details.....	1052
Polyline.....	1052
Fields details.....	1061
Popup.....	1061

---

projection.....	1063
projection.Cartesian.....	1063
projection.sphericalMercator.....	1064
projection.wgs84Mercator.....	1065
ready.....	1065
Rectangle.....	1067
Fields details.....	1076
regions.....	1076
regions.load.....	1076
RemoteObjectManager.....	1077
Fields details.....	1084
Events details.....	1084
Methods details.....	1084
route.....	1087
router.....	1089
router.addon.....	1089
router.Editor.....	1089
router.Path.....	1092
router.Route.....	1098
router.Segment.....	1104
router.ViaPoint.....	1107
router.WayPoint.....	1112
shape.....	1117
shape.Circle.....	1117
shape.LineString.....	1119
shape.MultiGeometry.....	1120
shape.MultiPolygon.....	1121
shape.Polygon.....	1123
shape.Rectangle.....	1124
shape.storage.....	1126
suggest.....	1126
SuggestView.....	1127
Fields details.....	1129
Events details.....	1130
Methods details.....	1130
template.....	1130
template.filtersStorage.....	1130
Template.....	1131
Methods details.....	1133
templateLayoutFactory.....	1133
Methods details.....	1134
traffic.....	1135
traffic.provider.....	1135
util.....	1145
util.AsyncStorage.....	1145
util.augment.....	1148
util.bind.....	1149
util.bounds.....	1150
util.cursor.....	1156
util.defineClass.....	1159
util.Dragger.....	1160
util.extend.....	1164
util.hd.....	1164
util.math.....	1166
util.pixelBounds.....	1167
util.requireCenterAndZoom.....	1171
util.Storage.....	1172
vow.....	1173

---

Methods details.....	1174
vow.Deferred.....	1175
vow.Promise.....	1177

# Yandex.Maps API Reference

This reference guide describes the JavaScript API version 2.1.72.

The release date is December 10, 2018.

## Fixed:

- Error occurred when deleting the map DOM-element.
- The incorrect traffic level was shown when two maps with the traffic layer enabled were created on the same page.
- Driving routes in the 'routePanel' were constructed without traffic allowances.
- Errors when creating a map in the hidden container.
- Layout fixes.

## Balloon

Extends [IBalloon](#), [Popup](#).

A balloon is a popup window that can display any HTML content. There is usually just one balloon instance on the map and it is managed via special managers (for example, [maps](#), [geo objects](#), [hotspot layers](#) and so on). Don't create them yourself, unless truly necessary.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
Balloon(map[, options])
```

### Parameters:

Parameter	Default value	Description
<a href="#">map</a> *	—	Type: <a href="#">Map</a> Reference to a map object.
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.autoPan</a>	true	Type: Boolean To move the map to show the opened balloon.
<a href="#">options.autoPanCheckZoomRange</a>	false	Type: Boolean Enables autoscaling when it is impossible to display the map after dragging on the same scale.
<a href="#">options.autoPanDuration</a>	500	Type: Number The duration of the movement to the point of the balloon (in milliseconds).

Parameter	Default value	Description
<a href="#">options.autoPanMargin</a>	34	Type: Number Number[]  Offset or offsets from the edges of the visible map area when executing <code>autoPan</code> . The value can be set as a single number (equal margins on all sides), as two numbers (for vertical and horizontal margins), or as four numbers (in the order of upper, right, lower, and left margins). Keep in mind that this value will be added to the value calculated in the margins manager <a href="#">map.margin.Manager</a> .
<a href="#">options.autoPanUseMapMargin</a>	true	Type: Boolean  Whether to account for map margins <a href="#">map.margin.Manager</a> when executing <code>autoPan</code> .
<a href="#">options.closeButton</a>	true	Type: Boolean  Flag for the Close button.
<a href="#">options.closeTimeout</a>	700	Type: Number  Delay before closing (in ms).
<a href="#">options.contentLayout</a>	—	Type: Function String  Layout for balloon content. (Type: constructor for an object with the <a href="#">ILayout</a> interface or the layout key).
<a href="#">options.interactivityModel</a>	—	Type: String  Key for the interactivity model. Available keys and their values are listed in the description of <a href="#">interactivityModel.storage</a> .
<a href="#">options.layout</a>	islands#balloon	Type: Function String  External layout for the balloon. (Type: constructor for an object with the <a href="#">ILayout</a> interface or the layout key).
<a href="#">options.maxHeight</a>	—	Type: Number  Maximum height, in pixels.
<a href="#">options.maxWidth</a>	—	Type: Number  Maximum width, in pixels.
<a href="#">options.minHeight</a>	—	Type: Number  Minimum height, in pixels.
<a href="#">options.minWidth</a>	—	Type: Number  Minimum width, in pixels.

Parameter	Default value	Description
<a href="#">options.offset</a>	—	Type: Number[]  Additional position offset relative to the anchor point.
<a href="#">options.openTimeout</a>	150	Type: Number  Delay before opening (in ms).
<a href="#">options.pane</a>	'balloon'	Type: String  Key of the pane that the balloon overlay is placed in.
<a href="#">options.panelContentLayout</a>	null	Type: Function String  The layout of the balloon contents in the panel mode. If this option is omitted, the value of the <code>contentLayout</code> option is used. (Type: constructor for an object with the <a href="#">ILayout</a> interface or the layout key).
<a href="#">options.panelMaxHeightRatio</a>	—	Type: Number  The maximum height of the balloon panel. Defined as the coefficient relative to the map height: a number from 0 to 1.
<a href="#">options.panelMaxMapArea</a>	—	Type: Number  The maximum area of the map at which the balloon will be displayed in the panel mode. You can disable panel mode by setting the value to 0, and vice versa, you can always show the balloon in panel mode by setting the value to <i>Infinity</i> .
<a href="#">options.shadow</a>	true	Type: Boolean  Flag for whether there is a shadow.
<a href="#">options.shadowLayout</a>	—	Type: Function String  Layout for the shadow. (Type: constructor for an object with the <a href="#">ILayout</a> interface or the layout key).
<a href="#">options.shadowOffset</a>	—	Type: Number[]  Additional position offset of the shadow relative to the anchor point.
<a href="#">options.zIndex</a>	—	Type: String  The z-index of the balloon.

\* Mandatory parameter/option.

#### Example:

```
// Creating an independent balloon instance and displaying it in the center of the map.
```

```
var balloon = new ymaps.Balloon(myMap);
// Here map options are set to parent options,
// where they contain default values for mandatory options.
balloon.options.setParent(myMap.options);
// Opening a balloon at the center of the map:
balloon.open(myMap.getCenter());
```

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">autopanbegin</a>	Start of automatic shifting of the map center initiated by the autoPan method. Instance of the <a href="#">Event</a> class.
<a href="#">autopanend</a>	End of automatic shifting of the map center initiated by the autoPan method. Instance of the <a href="#">Event</a> class.
<a href="#">beforeuserclose</a>	The event which precedes <a href="#">Balloon.event:userclose</a> . Allows you to cancel the user's action by calling the preventDefault method. Instance of the <a href="#">Event</a> class.
<a href="#">close</a>	Closing the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">open</a>	Opening the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">userclose</a>	Balloon closed by the user. Instance of the <a href="#">Event</a> class.

## Methods

Name	Returns	Description
<a href="#">autoPan()</a>	<a href="#">vow.Promise</a>	Moves the map so that the balloon is visible. Inherited from <a href="#">IBalloon</a> .
<a href="#">close([force])</a>	<a href="#">vow.Promise</a>	Closes the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">getData()</a>		Returns info object data. Inherited from <a href="#">IPopup</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object to return the overlay. Inherited from <a href="#">IPopup</a> .

Name	Returns	Description
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>	Returns the overlay, if one exists. Inherited from <a href="#">IPopup</a> .
<a href="#">getPosition()</a>		Returns the coordinates of the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">isOpen()</a>	Boolean	Returns the info object state: open/closed. Inherited from <a href="#">IPopup</a> .
<a href="#">open([position[, data]])</a>	<a href="#">vow.Promise</a>	Opens the info object at the specified position. If the info object is already open, it moves it to the specified point. The format and content of the coordinates is determined by the <a href="#">IProjection</a> that is in the options. Inherited from <a href="#">IPopup</a> .
<a href="#">setData(data)</a>	<a href="#">vow.Promise</a>	Defines new data for the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">setPosition(position)</a>	<a href="#">vow.Promise</a>	Specifies a new position for the info object. Inherited from <a href="#">IPopup</a> .

## Events details

### autopanbegin

Start of automatic shifting of the map center initiated by the autoPan method. Instance of the [Event](#) class.

### autopanend

End of automatic shifting of the map center initiated by the autoPan method. Instance of the [Event](#) class.

### beforeuserclose

The event which precedes [Balloon.event:userclose](#). Allows you to cancel the user's action by calling the preventDefault method. Instance of the [Event](#) class.

### userclose

Balloon closed by the user. Instance of the [Event](#) class.

## behavior

### behavior.DbClickZoom

Extends [IBehavior](#).

The "zooming the map with double-click" behavior.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

## Constructor

```
behavior.DblClickZoom([options])
```

### Parameters:

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.centering</a>	true	Type: Boolean If true, the map is zoomed on double-click so that the point under the mouse cursor becomes the map center; if false, the point under the mouse cursor stays in the same position when zooming on double-click.
<a href="#">options.duration</a>	200	Type: Number Duration of animation for zooming on double-click (0 - no animation).
<a href="#">options.useMapMargin</a>	true	Type: Boolean When centering, whether to consider margins that were calculated in the margins manager <a href="#">map.margin.Manager</a> .

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

### Events

Name	Description
<a href="#">disable</a>	Disabling behaviors. Inherited from <a href="#">IBehavior</a> .
<a href="#">enable</a>	Enabling behaviors. Inherited from <a href="#">IBehavior</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .

Name	Description
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>

## Methods

Name	Returns	Description
<a href="#">disable()</a>		<p>Disables the behavior.</p> <p>Inherited from <a href="#">IBehavior</a>.</p>
<a href="#">enable()</a>		<p>Enables the behavior.</p> <p>Inherited from <a href="#">IBehavior</a>.</p>
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	<p>Returns link to the parent object, or null if the parent element was not set.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>
<a href="#">isEnabled()</a>	Boolean	<p>Checks whether the behavior is enabled.</p> <p>Inherited from <a href="#">IBehavior</a>.</p>
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	<p>Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>

## behavior.Drag

Extends [IBehavior](#).

The "dragging the map using the mouse or single touch" behavior.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
behavior.Drag([options])
```

### Parameters:

Parameter	Default value	Description
<a href="#">options</a>	—	<p>Type: Object</p> <p>Options.</p>
<a href="#">options.actionCursor</a>	'grabbing'	<p>Type: String</p> <p>Cursor for the behavior <a href="#">behavior.Drag</a> when dragging the map.</p>

Parameter	Default value	Description
<a href="#">options.cursor</a>	'grab'	Type: String  Cursor for the behavior <a href="#">behavior.Drag</a> when pointing at the map.
<a href="#">options.inertia</a>	true	Type: Boolean  Enables kinetic inertia at the end of dragging.
<a href="#">options.inertiaDuration</a>	400	Type: Number String  Duration of inertia, in ms. The "auto" string value sets the duration of inertia proportional to the distance.
<a href="#">options.tremor</a>	2	Type: Integer  Minimal cursor movement after pressing the mouse button, before the map begins to move.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

### Events

Name	Description
<a href="#">disable</a>	Disabling behaviors. Inherited from <a href="#">IBehavior</a> .
<a href="#">enable</a>	Enabling behaviors. Inherited from <a href="#">IBehavior</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .

**Methods**

Name	Returns	Description
<a href="#">disable()</a>		Disables the behavior. Inherited from <a href="#">IBehavior</a> .
<a href="#">enable()</a>		Enables the behavior. Inherited from <a href="#">IBehavior</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">isEnabled()</a>	Boolean	Checks whether the behavior is enabled. Inherited from <a href="#">IBehavior</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IChildOnMap</a> .

**behavior.LeftMouseButtonMagnifier**

Extends [IBehavior](#).

The "zooming the map when selecting an area with the left mouse button" behavior.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
behavior.LeftMouseButtonMagnifier([options])
```

Creates the "zooming the map when selecting an area with the left mouse button" behavior.

**Parameters:**

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.actionCursor</a>	'crosshair'	Type: String The cursor when selecting the area to magnify with the enabled behavior <a href="#">behavior.LeftMouseButtonMagnifier</a> . The list of all available cursors is provided in the <a href="#">cursors manager</a> .
<a href="#">options.cursor</a>	'zoom'	Type: String The cursor for the enabled behavior <a href="#">behavior.LeftMouseButtonMagnifier</a> . The list of all available cursors is provided in the <a href="#">cursors manager</a> .

Parameter	Default value	Description
<a href="#">options.duration</a>	300	Type: Number  The duration of magnification animation when using the behavior <a href="#">behavior.LeftMouseButtonMagnifier</a> , in ms.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">disable</a>	Disabling behaviors. Inherited from <a href="#">IBehavior</a> .
<a href="#">enable</a>	Enabling behaviors. Inherited from <a href="#">IBehavior</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li><code>oldParent</code> - Old parent.</li> <li><code>newParent</code> - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .

## Methods

Name	Returns	Description
<a href="#">disable()</a>		Disables the behavior. Inherited from <a href="#">IBehavior</a> .
<a href="#">enable()</a>		Enables the behavior. Inherited from <a href="#">IBehavior</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">isEnabled()</a>	Boolean	Checks whether the behavior is enabled. Inherited from <a href="#">IBehavior</a> .

Name	Returns	Description
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .

## behavior.MultiTouch

Extends [IBehavior](#).

The "zooming the map using multitouch" behavior.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
behavior.MultiTouch([options])
```

### Parameters:

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object  Options.
<a href="#">options.tremor</a>	2	Type: Number  The minimal movement on the device screen (in pixels) to trigger the behavior <a href="#">behavior.MultiTouch</a> .

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager.  Inherited from <a href="#">ICustomizable</a> .

### Events

Name	Description
<a href="#">disable</a>	Disabling behaviors.  Inherited from <a href="#">IBehavior</a> .
<a href="#">enable</a>	Enabling behaviors.  Inherited from <a href="#">IBehavior</a> .
<a href="#">optionschange</a>	Change to the object options.  Inherited from <a href="#">ICustomizable</a> .

Name	Description
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>

## Methods

Name	Returns	Description
<a href="#">disable()</a>		<p>Disables the behavior.</p> <p>Inherited from <a href="#">IBehavior</a>.</p>
<a href="#">enable()</a>		<p>Enables the behavior.</p> <p>Inherited from <a href="#">IBehavior</a>.</p>
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	<p>Returns link to the parent object, or null if the parent element was not set.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>
<a href="#">isEnabled()</a>	Boolean	<p>Checks whether the behavior is enabled.</p> <p>Inherited from <a href="#">IBehavior</a>.</p>
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	<p>Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>

## behavior.RightMouseButtonMagnifier

Extends [IBehavior](#).

The "zooming the map when selecting an area with the right mouse button" behavior.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
behavior.RightMouseButtonMagnifier([options])
```

Creates the "zooming the map when selecting an area with the right mouse button" behavior.

### Parameters:

Parameter	Default value	Description
<a href="#">options</a>	—	<p>Type: Object</p> <p>Options.</p>

Parameter	Default value	Description
<a href="#">options.actionCursor</a>	'crosshair'	Type: String  The cursor when selecting the area to magnify with the enabled behavior <a href="#">behavior.RightMouseButtonMagnifier</a> . The list of all available cursors is provided in the <a href="#">cursors manager</a> .
<a href="#">options.duration</a>	300	Type: Number  The duration of animation when selecting an area using the behavior <a href="#">behavior.RightMouseButtonMagnifier</a> , in ms.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

### Events

Name	Description
<a href="#">disable</a>	Disabling behaviors. Inherited from <a href="#">IBehavior</a> .
<a href="#">enable</a>	Enabling behaviors. Inherited from <a href="#">IBehavior</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .

### Methods

Name	Returns	Description
<a href="#">disable()</a>		Disables the behavior. Inherited from <a href="#">IBehavior</a> .
<a href="#">enable()</a>		Enables the behavior. Inherited from <a href="#">IBehavior</a> .

Name	Returns	Description
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IChildOnMap</a> .
<a href="#">isEnabled()</a>	Boolean	Checks whether the behavior is enabled.  Inherited from <a href="#">IBehavior</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .

## behavior.RouteEditor

Extends [IBehavior](#).

"Route editor" behavior.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
behavior.RouteEditor()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager.  Inherited from <a href="#">ICustomizable</a> .

### Events

Name	Description
<a href="#">disable</a>	Disabling behaviors.  Inherited from <a href="#">IBehavior</a> .
<a href="#">enable</a>	Enabling behaviors.  Inherited from <a href="#">IBehavior</a> .
<a href="#">optionschange</a>	Change to the object options.  Inherited from <a href="#">ICustomizable</a> .

Name	Description
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .
<a href="#">routechange</a>	A changed route event resulting from calling the <code>setState</code> method or enabling/disabling behaviors. You can get the old and new routes from the properties of the "oldRoute" and "newRoute" events, respectively.

## Methods

Name	Returns	Description
<a href="#">disable()</a>		Disables the behavior. Inherited from <a href="#">IBehavior</a> .
<a href="#">enable()</a>		Enables the behavior. Inherited from <a href="#">IBehavior</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">getRoute()</a>	<a href="#">router.Route</a>	Returns route.
<a href="#">getState()</a>	String	Returns the current state of the route editor in encoded format.
<a href="#">isEnabled()</a>	Boolean	Checks whether the behavior is enabled. Inherited from <a href="#">IBehavior</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">setState(state)</a>		Restores the state of the route editor from the encoded string. When setting a new route, the "routechange" event is called.

## Events details

### routechange

A changed route event resulting from calling the `setState` method or enabling/disabling behaviors. You can get the old and new routes from the properties of the "oldRoute" and "newRoute" events, respectively.

## Methods details

### getRoute

```
{router.Route} getRoute()
```

Returns route.

### getState

```
{String} getState()
```

Returns the current state of the route editor in encoded format.

### setState

```
{ } setState(state)
```

Restores the state of the route editor from the encoded string. When setting a new route, the "routechange" event is called.

#### Parameters:

Parameter	Default value	Description
<code>state *</code>	—	<p>Type: String null</p> <p>Encoded state of the route editor. The state is set in the following format: <code>rt=point1~point2~point3~point4...&amp;via=via-point-indexes</code> For example, if the encoded string looks like this: <code>"rt=50,30~42,35~45,32~40,30&amp;via=1,2"</code>, it means the waypoints on the route should be "50,30" and "40,30", and the midpoints are "42,35" and "45,32". If the "via" parameter is omitted, the route will consist solely of waypoints.</p>

\* Mandatory parameter/option.

#### Example:

```
var behavior = map.behaviors.get('routeEditor');
behavior.events.add('routechange', function (e) {
  var newRoute = e.get('newRoute');
  alert(newRoute.getLength()); // length of the new route
  setTimeout(function () {
    // deleting the route
    behavior.setState(null);
  });
});
behavior.setState('rt=55.874872,37.562677~55.92517867214157,37.62725433916199~55.920011490602526,37.6629269628905&via=1');
```

## behavior.Ruler

Extends [IBehavior](#).

The "Ruler" behavior. For marking points on the map and displaying the distance between them.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
behavior.Ruler([options])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object  Options.
<a href="#">options.balloonAutoPan</a>	true	Type: Boolean  Whether to auto-position the map when opening the ruler balloon.
<a href="#">options.balloonAutoPanUseMapMargin</a>	true	Type: Boolean  Whether to account for map margins <a href="#">map.margin.Manager</a> when executing <code>autoPan</code> for the ruler balloon.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">geometry.LineString</a>	"Line" behavior geometry.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager.  Inherited from <a href="#">ICustomizable</a> .

**Events**

Name	Description
<a href="#">disable</a>	Disabling behaviors.  Inherited from <a href="#">IBehavior</a> .
<a href="#">enable</a>	Enabling behaviors.  Inherited from <a href="#">IBehavior</a> .
<a href="#">optionschange</a>	Change to the object options.  Inherited from <a href="#">ICustomizable</a> .
<a href="#">parentchange</a>	The parent object reference changed.  Data fields: <ul style="list-style-type: none"> <li><code>oldParent</code> - Old parent.</li> <li><code>newParent</code> - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .

**Methods**

Name	Returns	Description
<a href="#">close()</a>	Boolean	Deletes all the points on the ruler. If the current number of points is more than two, confirmation of this action will be requested.
<a href="#">disable()</a>		Disables the behavior. Inherited from <a href="#">IBehavior</a> .
<a href="#">enable()</a>		Enables the behavior. Inherited from <a href="#">IBehavior</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">getState()</a>	String	The ruler state is described by a string consisting of sequences separated by the "~" symbol. Each sequence is a substring in the format "longitude,latitude" that describes the increment in coordinates relative to the previous ruler point.
<a href="#">isEnabled()</a>	Boolean	Checks whether the behavior is enabled. Inherited from <a href="#">IBehavior</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">setState(state)</a>		Retrieves the ruler state from the encoded string. See <a href="#">behavior.Ruler.getState</a> .

**Fields details****geometry**

```
{geometry.LineString} geometry
```

"Line" behavior geometry.

**Example:**

```
// Setting coordinates of the first point on the ruler.
myMap.behaviors.get('ruler').geometry.set(0, [0, 0]);
```

## Methods details

### close

```
{Boolean} close()
```

Deletes all the points on the ruler. If the current number of points is more than two, confirmation of this action will be requested.

**Returns** true, if the action was completed successfully.

### getState

```
{String} getState()
```

The ruler state is described by a string consisting of sequences separated by the "~" symbol. Each sequence is a substring in the format "longitude,latitude" that describes the increment in coordinates relative to the previous ruler point.

**Returns** the current state of the ruler, in encoded format.

### setState

```
{ } setState(state)
```

Retrieves the ruler state from the encoded string. See `behavior.Ruler.getState`.

#### Parameters:

Parameter	Default value	Description
<code>state</code> *	—	Type: String Encoded state of the ruler.

\* Mandatory parameter/option.

## behavior.ScrollZoom

Extends [IBehavior](#).

The "zooming the map with the mouse wheel" behavior.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
behavior.ScrollZoom([options])
```

#### Parameters:

Parameter	Default value	Description
<code>options</code>	—	Type: Object Options.

Parameter	Default value	Description
<a href="#">options.maximumDelta</a>	5	Type: Number  The maximum change in the map zoom for a single scroll event (uninterrupted turn of the mouse wheel). When this limit is reached, the map is re-drawn (tiles are shown for the zoom level that was reached).
<a href="#">options.speed</a>	5	Type: Number  The speed of zooming the map with the enabled behavior <a href="#">behavior.ScrollZoom</a> , in zoom levels per second.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

### Events

Name	Description
<a href="#">disable</a>	Disabling behaviors. Inherited from <a href="#">IBehavior</a> .
<a href="#">enable</a>	Enabling behaviors. Inherited from <a href="#">IBehavior</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .

### Methods

Name	Returns	Description
<a href="#">disable()</a>		Disables the behavior. Inherited from <a href="#">IBehavior</a> .
<a href="#">enable()</a>		Enables the behavior. Inherited from <a href="#">IBehavior</a> .

Name	Returns	Description
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IChildOnMap</a> .
<a href="#">isEnabled()</a>	Boolean	Checks whether the behavior is enabled.  Inherited from <a href="#">IBehavior</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .

## behavior.storage

Static object.

Instance of [util.Storage](#)

Storage for map behavior classes. A new behavior is added to the map via this storage.

By default, the following behaviors are added to the storage:

- "drag" - Dragging the map when the left mouse button is held down, or by a single touch [behavior.Drag](#).
- "scrollZoom" - Changing the zoom with the mouse wheel [behavior.ScrollZoom](#) (only for desktop browsers).
- "dblClickZoom" - Zooming the map on a double click [behavior.DblClickZoom](#).
- "multiTouch" - Zooming the map with a multi touch (i.e. on a touch screen) [behavior.MultiTouch](#) (only for mobile browsers).
- "rightMouseButtonMagnifier" - Magnifying the area that is selected using the right mouse button (for desktop browsers only), [behavior.RightMouseButtonMagnifier](#) (for desktop browsers only).
- "leftMouseButtonMagnifier" - Magnifying the area selected by the left mouse button or a single touch, [behavior.LeftMouseButtonMagnifier](#).
- "ruler" - Measuring distance [behavior.Ruler](#).
- "routeEditor" - Route editor [behavior.RouteEditor](#).

### Methods

#### Methods

Name	Returns	Description
<a href="#">add(key, object)</a>	<a href="#">util.Storage</a>	Adds an object to storage.
<a href="#">get(key)</a>	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
<a href="#">remove(key)</a>	<a href="#">util.Storage</a>	Deletes the "key: value" pair from storage.

## borders

### borders.load

Static function.

Provides access to the geometry of various regions and countries. If you are using a content security policy, then you need to add the 'connect-src' rule for the following hosts: <https://api-maps.yandex.ru> <https://suggest-maps.yandex.ru> [https://\\*.maps.yandex.net](https://*.maps.yandex.net) <https://yandex.ru>

**Returns** Promise object.

```
{ vow.Promise } borders.load(region[, options])
```

### Parameters:

Parameter	Default value	Description
<code>region</code> *	—	Type: String  The ISO_3166-1 country code (RU, UA, BY, KZ, TR) for loading the regional area, '001' for loading the geometry of country borders, or AQ for loading the geometry of Antarctic.
<code>options</code>	—	Type: Object  Display options.
<code>options.disputedBorders</code>	—	Type: String  Two-letter code of the country to use as the official reference for determining the administrative subordination of disputed territories. Accepted values: 'RU', 'UA', 'UN'. By default, it coincides with the country code that is specified when loading the API. Unsupported country codes are reset to RU. For the region '001' (borders of countries), the code 'UN' is supported — world borders according to the United Nations.
<code>options.lang</code>	—	Type: String  Language (ru, en, uk, be, kk, tr).
<code>options.quality</code>	1	Type: Number  Quality level. Available values: <ul style="list-style-type: none"> <li>• 0 - minimal quality</li> <li>• 1 - standard quality</li> <li>• 2 - improved quality</li> <li>• 3 - high quality</li> </ul> The quality level affects how accurately curves are represented, as well as the volume of the data file.

\* Mandatory parameter/option.

### Examples:

#### 1.

```
// Show objects on the map using ObjectManager
ymaps.borders.load('RU', {
  lang: 'en'
}).then(function (geojson) {
  var features = geojson.features.map(function (feature) {
    feature.id = feature.properties.iso3166;
    return feature;
  });
});
```

```
});  
var objectManager = new ymaps.ObjectManager();  
objectManager.add(features);  
myMap.geoObjects.add(objectManager);  
});
```

## 2.

```
// Show objects on the map using GeoObject  
ymaps.borders.load('RU', {  
  lang: 'en'  
}).then(function (geojson) {  
  for (var i = 0; i < geojson.features.length; i++) {  
    var geoObject = new ymaps.GeoObject(geojson.features[i]);  
    myMap.geoObjects.add(geoObject);  
  }  
});
```

## Circle

Extends [GeoObject](#).

Circle. A geo object with the geometry [geometry.Circle](#).

See [GeoObject geometry.Circle](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
Circle(geometry[, properties[, options]])
```

Creates an instance of a circle.

### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">ICircleGeometry</a>   <a href="#">Number[][]</a>   Object  Reference to a point geometry object or an array, in which the first item is the coordinates of the center of the circle, and the second is the radius in meters, or an object with the parameters of the geometry.

Parameter	Default value	Description
<a href="#">properties</a>	—	<p>Type: Object <a href="#">IDataManager</a></p> <p>Circle data. Corresponds to the data for the <a href="#">GeoObject</a> object. Can be set as an instance of a class that implements the <a href="#">IDataManager</a> interface, or as a hash. When options are set to default values, the following data fields are interpreted by a circle:</p> <ul style="list-style-type: none"> <li>hintContent - Content of the circle's popup hint.</li> <li>balloonContent - Content of the circle's balloon.</li> <li>balloonContentHeader - Content of the circle balloon title.</li> <li>balloonContentBody - Content of the main part of the circle's balloon.</li> <li>balloonContentFooter - Content of the lower part of the circle's balloon.</li> </ul> <p>The balloonContent field is a shortcut for the balloonContentBody field, but if they are both set simultaneously, balloonContentBody takes priority. You can also add your own custom fields to the circle data and use them wherever possible. For example, in the circle layout or balloon layout.</p>
<a href="#">options</a>	—	<p>Type: Object</p> <p>Circle options. Using this parameter, you can set options for the circle itself, as well as for its parts:</p> <ul style="list-style-type: none"> <li>Options for the circle's <a href="#">balloon</a> with the <code>balloon</code> prefix.</li> <li>Options for the circle's <a href="#">popup</a> hint with the <code>hint</code> prefix.</li> <li>Options for the circle's geometry editor with the <code>editor</code> prefix. See the description of the <a href="#">geometryEditor.Circle</a> class.</li> <li>Geometry options can be set without a prefix. See the description of the <a href="#">IGeometry</a> class for the <a href="#">geometry.Circle</a> geometry.</li> </ul>
<a href="#">options.circleOverlay</a>	"default#circle"	<p>Type: String Function</p> <p>Key identifier from <a href="#">overlay.storage</a> or the overlay class. The generator function accepts three parameters:</p> <ul style="list-style-type: none"> <li>geometry: <a href="#">IPixelCircleGeometry</a> - The pixel geometry itself.</li> <li>data: Object - The overlay data.</li> <li>options: Object - The overlay options.</li> </ul> <p>And returns <a href="#">vow.Promise</a>.</p>

Parameter	Default value	Description
<a href="#">options.cursor</a>	"pointer"	Type: String  Type of cursor over a circle.
<a href="#">options.draggable</a>	false	Type: Boolean  Checks whether the circle can be dragged.
<a href="#">options.fill</a>	true	Type: Boolean  Whether the shape is filled.
<a href="#">options.fillColor</a>	"0066ff99"	Type: String  Fill color.
<a href="#">options.fillImageHref</a>	—	Type: String  Background image. When this option is enabled in <b>stretch</b> mode, the "fillColor" value is ignored.
<a href="#">options.fillMethod</a>	'stretch'	Type: String  Type of background fill. Accepts one of two values: <ul style="list-style-type: none"> <li>stretch - The background image stretches to fit the size of the overlay.</li> <li>tile - The background image is repeated, without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.</li> </ul>
<a href="#">options.fillOpacity</a>	1	Type: Number  Fill transparency.
<a href="#">options.hasBalloon</a>	true	Type: Boolean  Checks whether the circle has the "balloon" field.
<a href="#">options.hasHint</a>	true	Type: Boolean  Checks whether the circle has the "hint" field.
<a href="#">options.hideIconOnBalloonOpen</a>	true	Type: Boolean  Hide the icon when opening the balloon.
<a href="#">options.interactiveZIndex</a>	false	Type: Boolean  Enables automatically modifying the z-index of the circle depending on its state.

Parameter	Default value	Description
<a href="#">options.interactivityModel</a>	"default#geoObject"	Type: String  Interactivity model. Available keys and their values are listed in the description of <a href="#">interactivityModel.storage</a> .
<a href="#">options.opacity</a>	1	Type: Number  Transparency.
<a href="#">options.openBalloonOnClick</a>	true	Type: Boolean  Checks whether to show the balloon when the circle is clicked on.
<a href="#">options.openEmptyBalloon</a>	false	Type: Boolean  Checks whether to show an empty balloon when the circle is clicked on.
<a href="#">options.openEmptyHint</a>	false	Type: Boolean  Checks whether to show an empty hint when the mouse pointer hovers over the circle.
<a href="#">options.openHintOnHover</a>	true	Type: Boolean  Checks whether to show the hint when the mouse pointer hovers over the circle.
<a href="#">options.outline</a>	true	Type: Boolean  Whether the circle has an outline.
<a href="#">options.pane</a>	"areas"	Type: String  The key of the pane where the circle overlay is placed.
<a href="#">options.strokeColor</a>	"0066ffff"	Type: String String[]  Color of the line or outline. You can set multiple values for a multistroke outline.
<a href="#">options.strokeOpacity</a>	1	Type: Number Number[]  Transparency of the line or outline. You can set multiple values for a multistroke outline.
<a href="#">options.strokeStyle</a>	—	Type: String Object String[] Object[]  Style of the line or outline. You can set multiple values for a multistroke outline.
<a href="#">options.strokeWidth</a>	1	Type: Number Number[]  Thickness of the line or outline. You can set multiple values for a multistroke outline.

Parameter	Default value	Description
<a href="#">options.syncOverlayInit</a>	false	Type: Boolean  Enables synchronously adding an overlay to the map. By default, overlays are added to the map asynchronously to prevent the browser from hanging when adding a large number of geo objects. However, adding asynchronously does not allow accessing the overlay immediately after adding a circle to the map.
<a href="#">options.useMapMarginInDragging</a>	true	Type: Boolean  When an object is dragged to the edge of the map, the map center changes automatically. Whether to use map margins when automatically shifting the map center with <a href="#">map.margin.Manager</a> .
<a href="#">options.visible</a>	true	Type: Boolean  Checks circle visibility.
<a href="#">options.zIndex</a>	—	Type: Number  The z-index of a circle in its normal state. Lowest priority.
<a href="#">options.zIndexActive</a>	—	Type: Number  The z-index of a circle with an open balloon. Highest priority.
<a href="#">options.zIndexDrag</a>	—	Type: Number  The z-index of a circle that is being dragged.
<a href="#">options.zIndexHover</a>	—	Type: Number  The z-index of a circle when the mouse pointer is hovering over it.

\* Mandatory parameter/option.

#### Example:

```
// Creating a geodesic circle with a radius of 1000 kilometers.
var circle = new ymaps.Circle([[50, 75], 1000000], {}, {
  geodesic: true
});
// Adding the circle to the map.
myMap.geoObjects.add(circle);
```

#### Fields

Name	Type	Description
<a href="#">balloon</a>	<a href="#">geoObject.Balloon</a>	Balloon for a geo object.  Inherited from <a href="#">GeoObject</a> .
<a href="#">editor</a>	<a href="#">geometryEditor.Circle</a>	The "Circle" geometry editor.

Name	Type	Description
<a href="#">events</a>	<a href="#">event.Manager</a>	Event manager. Inherited from <a href="#">GeoObject</a> .
<a href="#">geometry</a>	<a href="#">geometry.Circle</a>	The "Circle" type of geometry.
<a href="#">hint</a>	<a href="#">geoObject.Hint</a>	Geo object hint. Inherited from <a href="#">GeoObject</a> .
<a href="#">indices</a>	ArrayBuffer	
<a href="#">options</a>	<a href="#">option.Manager</a>	Geo object options manager. Inherited from <a href="#">GeoObject</a> .
<a href="#">properties</a>	<a href="#">data.Manager</a>	Geo object data manager. Inherited from <a href="#">GeoObject</a> .
<a href="#">state</a>	<a href="#">data.Manager</a>	State of the geo object. Defined by the following fields: <ul style="list-style-type: none"> <li>• active: Boolean - Indicates that a balloon is open on the geo object.</li> <li>• hover: Boolean - Indicates that the mouse is currently pointed at the geo object.</li> <li>• drag: Boolean - Indicates that the geo object is being dragged</li> </ul> Inherited from <a href="#">GeoObject</a> .
<a href="#">vertices</a>	ArrayBuffer	

## Events

Name	Description
<a href="#">balloonclose</a>	Closing the balloon. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">GeoObject</a> .
<a href="#">balloonopen</a>	Opening a balloon on a geo object. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">GeoObject</a> .
<a href="#">beforedrag</a>	Event preceding the "drag" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>• position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• pixelOffset - Array of two numbers that describe the pixel offset at this step.</li> <li>• domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> Names of methods that are accessible via <a href="#">Event.callMethod</a> : <ul style="list-style-type: none"> <li>• setPixelOffset - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.</li> </ul> If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent drag event will be canceled. Inherited from <a href="#">GeoObject</a> .

Name	Description
<a href="#">beforedragstart</a>	<p>Event preceding the "dragstart" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">drag</a>	<p>Dragging a geo object. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>pixelOffset - Array of two numbers that describe the pixel offset at this step.</li> <li>domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">dragend</a>	<p>End of geo object dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">dragstart</a>	<p>Start of geo object dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">editorstatechange</a>	<p>Change in the state of the editor for the geo object's geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>originalEvent - Original event of the geometry editor.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>

Name	Description
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li></ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">hintclose</a>	<p>Closing the hint. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">hintopen</a>	<p>Opening a hint on a geo object. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>oldMap - Old map.</li><li>newMap - New map.</li></ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IChildOnMap</a> .

**Fields details****editor**

```
{geometryEditor.Circle} editor
```

The "Circle" geometry editor.

**geometry**

```
{geometry.Circle} geometry
```

The "Circle" type of geometry.

**indices**

```
{ArrayBuffer} indices
```

**vertices**

```
{ArrayBuffer} vertices
```

## clusterer

### clusterer.addon

#### clusterer.addon.balloon

**Note:** The constructor of the `clusterer.addon.balloon` class is hidden, as this class is not intended for autonomous initialization.

Static object.

[Methods](#)

#### Methods

Name	Returns	Description
<a href="#">get(clusterer)</a>	<a href="#">IPopupManager</a>	Returns the clusterer's balloon manager.

#### Methods details

##### get

```
{IPopupManager} get (clusterer)
```

**Returns** the clusterer's balloon manager.

#### Parameters:

Parameter	Default value	Description
<a href="#">clusterer</a> *	—	Type: <a href="#">Clusterer</a> Clusterer

\* Mandatory parameter/option.

#### Example:

```
ymaps.clusterer.addon.balloon.get(clusterer)
```

#### clusterer.addon.hint

**Note:** The constructor of the `clusterer.addon.hint` class is hidden, as this class is not intended for autonomous initialization.

Static object.

[Methods](#)

#### Methods

Name	Returns	Description
<a href="#">get(clusterer)</a>	<a href="#">IPopupManager</a>	Returns the clusterer's hint manager.

## Methods details

### get

```
{IPopupManager} get (clusterer)
```

Returns the clusterer's hint manager.

#### Parameters:

Parameter	Default value	Description
<code>clusterer *</code>	—	Type: <a href="#">Clusterer</a> Clusterer

\* Mandatory parameter/option.

#### Example:

```
ymaps.clusterer.addon.hint.get (clusterer)
```

## clusterer.Balloon

Extends [IBalloonManager](#).

The clusterer's balloon manager. Provides management of the cluster's balloon - opening it and hiding it. It uses the map balloon manager [map.Balloon](#). Clusterers contain an instance of this class, which is available as `myGeoObject.balloon`. Don't create new instances of this class unless necessary.

See [Balloon](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
clusterer.Balloon (clusterer)
```

#### Parameters:

Parameter	Default value	Description
<code>clusterer *</code>	—	Type: <a href="#">Clusterer</a> Clusterer.

\* Mandatory parameter/option.

### Fields

Name	Type	Description
<code>close</code>		Closing the balloon. <ul style="list-style-type: none"> <li>target - Link to the clusterer balloon manager.</li> <li>cluster - Reference to the cluster object.</li> </ul> Instance of the Event class.
<code>events</code>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

Name	Type	Description
<a href="#">open</a>		<p>Opening a balloon on a cluster. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>target - Link to the clusterer balloon manager.</li> <li>cluster - Reference to the cluster object.</li> </ul> <p>Instance of the Event class.</p>

## Events

Name	Description
<a href="#">autopanbegin</a>	<p>Start of automatic shifting of the map center initiated by the autoPan method. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>target - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> <p>Inherited from <a href="#">IBalloonManager</a>.</p>
<a href="#">autopanend</a>	<p>End of automatic shifting of the map center initiated by the autoPan method. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>target - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> <p>Inherited from <a href="#">IBalloonManager</a>.</p>
<a href="#">beforeuserclose</a>	<p>The event which precedes <a href="#">Balloon.event:userclose</a>. Allows you to cancel the user's action by calling the preventDefault method. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>target - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> <p>Inherited from <a href="#">IBalloonManager</a>.</p>
<a href="#">close</a>	<p>Closing the info object. Names of fields available via <a href="#">Event.get</a>:</p> <ul style="list-style-type: none"> <li>target - Reference to the object where the closing occurred.</li> </ul> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">open</a>	<p>Opening the info object. Names of fields available via <a href="#">Event.get</a>:</p> <ul style="list-style-type: none"> <li>target - Reference to the object where the opening occurred.</li> </ul> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">userclose</a>	<p>Balloon closed by the user. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>target - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> <p>Inherited from <a href="#">IBalloonManager</a>.</p>

## Methods

Name	Returns	Description
<a href="#">autoPan()</a>	<a href="#">vow.Promise</a>	<p>Moves the map so that the balloon is visible.</p> <p>Inherited from <a href="#">IBalloonManager</a>.</p>

Name	Returns	Description
<a href="#">destroy()</a>		Disables the info object manager.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">getData()</a>	Object null	Returns the data of the info object or 'null'.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOptions()</a>	<a href="#">IOptionManager</a>  null	Returns the options manager or 'null'.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object to return the overlay.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	Returns the overlay, if one exists.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">getPosition()</a>	Number[][] null	Returns the coordinates of the info object or 'null'.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">isOpen()</a>	Boolean	Returns the info object state: open/closed.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">setData(data)</a>	<a href="#">vow.Promise</a>	Defines new data for the info object.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">setOptions(options)</a>	<a href="#">vow.Promise</a>	Defines new options for the info object.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">setPosition(position)</a>	<a href="#">vow.Promise</a>	Specifies a new position for the info object.  Inherited from <a href="#">IPopupManager</a> .

### Fields details

#### close

```
close
```

Closing the balloon.

- target - Link to the clusterer balloon manager.
- cluster - Reference to the cluster object.

Instance of the Event class.

### open

```
open
```

Opening a balloon on a cluster. Names of fields that are available via the [Event.get](#) method:

- target - Link to the clusterer balloon manager.
- cluster - Reference to the cluster object.

Instance of the Event class.

### Example:

```
clusterer.balloon.events.add('open', function (e) {
    var clusterPlacemark = e.get('cluster');
});
```

## clusterer.Hint

Extends [IHintManager](#).

The clusterer's hint manager. Provides management of the cluster's hint, opening it and hiding it. It uses the map hint manager [map.Hint](#) inside itself. Clusterers contain an instance of this class, which is available as `myClusterer.hint`. Don't create new instances of this class unless necessary.

See [Hint](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
clusterer.Hint(clusterer)
```

### Parameters:

Parameter	Default value	Description
<code>clusterer</code> *	—	Type: <a href="#">Clusterer</a> Clusterer.

\* Mandatory parameter/option.

### Fields

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

### Events

Name	Description
<code>close</code>	Closing the info object. Names of fields available via <a href="#">Event.get</a> : <ul style="list-style-type: none"> <li>• target - Reference to the object where the closing occurred.</li> </ul> Inherited from <a href="#">IPopupManager</a> .

Name	Description
<a href="#">open</a>	<p>Opening the info object. Names of fields available via <a href="#">Event.get</a>:</p> <ul style="list-style-type: none"> <li>target - Reference to the object where the opening occurred.</li> </ul> <p>Inherited from <a href="#">IPopupManager</a>.</p>

## Methods

Name	Returns	Description
<a href="#">close</a> ([force])	<a href="#">vow.Promise</a>	<p>Closes the info object.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">destroy</a> ()		<p>Disables the info object manager.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">getData</a> ()	Object null	<p>Returns the data of the info object or 'null'.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">getOptions</a> ()	<a href="#">IOptionManager</a>  null	<p>Returns the options manager or 'null'.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">getOverlay</a> ()	<a href="#">vow.Promise</a>	<p>Returns the promise object to return the overlay.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">getOverlaySync</a> ()	<a href="#">IOverlay</a>  null	<p>Returns the overlay, if one exists.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">getPosition</a> ()	Number[] null	<p>Returns the coordinates of the info object or 'null'.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">isOpen</a> ()	Boolean	<p>Returns the info object state: open/closed.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">open</a> ([position[, data[, options]]])	<a href="#">vow.Promise</a>	<p>Opens the info object at the specified position.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">setData</a> (data)	<a href="#">vow.Promise</a>	<p>Defines new data for the info object.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>

Name	Returns	Description
<a href="#">setOptions(options)</a>	<a href="#">vow.Promise</a>	Defines new options for the info object.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">setPosition(position)</a>	<a href="#">vow.Promise</a>	Specifies a new position for the info object.  Inherited from <a href="#">IPopupManager</a> .

## Clusterer

Extends [IChildOnMap](#), [ICustomizable](#), [IEventEmitter](#), [IParentOnMap](#).

Geo object clusterer. Clusterizes objects in the visible area of the map. If the object does not fall within the visible area of the map, it will not be added to the map. Note, that the clusterer does not react to changing the coordinates of objects (either programmatically, or as the result of dragging). If you want to change the coordinates of some object in the clusterer, you should first delete the object from the clusterer and then add it back.

See [ClusterPlacemark](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
Clusterer([options])
```

### Parameters:

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object  Options. Options for child cluster objects are set with the "cluster" prefix. See <a href="#">ClusterPlacemark</a> .
<a href="#">options.gridSize</a>	64	Type: Number  The size of cluster cells, in pixels. The value must be equal $2^n$ (an area of 256 by 256 pixels should fit an even number of cells).
<a href="#">options.groupByCoordinates</a>	false	Type: Boolean  Special operation mode of the clusterer allowing clusters to be formed only from geo objects with the same coordinates.
<a href="#">options.hasBalloon</a>	true	Type: Boolean  Flag whether the clusterer has the <code>.balloon</code> field. If a balloon doesn't need to be opened when clicking the cluster, we recommend setting this option to the "false" value to avoid unnecessary initializations.

Parameter	Default value	Description
<a href="#">options.hasHint</a>	true	Type: Boolean  Flag whether the clusterer has the <code>.hint</code> field. If a popup hint doesn't need to be displayed when the cluster is pointed at, we recommend setting this option to the "false" value to avoid unnecessary initializations.
<a href="#">options.margin</a>	10	Type: Number Number[]  Number or array of numbers that set the offset for the cluster center relative to the clusterization cells. If a single number is set, it is applied to each side. If two numbers are set, they are the vertical and horizontal offsets, respectively. If an array of four numbers is set, they are the top, right, bottom, and left margins.
<a href="#">options.maxZoom</a>	Infinity	Type: Number[]  The maximum map zoom that object clusterization occurs at. Even if clusterization is disabled, only objects in the visible map area will be shown.
<a href="#">options.minClusterSize</a>	2	Type: Number  Minimum number of objects to make up a cluster.
<a href="#">options.preset</a>	—	Type: String  Key for the clusterer's preset options. A list of keys available in the <code>package.clusters</code> package can be found in the description of <a href="#">option.presetStorage</a> .
<a href="#">options.showInAlphabeticalOrder</a>	false	Type: Boolean  Show placemarks in the balloon in alphabetical order when the cluster is clicked on. The cluster's geo objects are sorted by special fields in the data of these geo objects - <code>clusterCaption</code> (or <code>balloonContentHeader</code> , if the previous field is not defined). By default, geo objects are shown in the order in which they were added to the clusterer.
<a href="#">options.useMapMargin</a>	true	Type: Boolean  Whether to account for map margins <a href="#">map.margin.Manager</a> when zooming the map in after a click on a cluster.

Parameter	Default value	Description
<a href="#">options.viewportMargin</a>	128	Type: Number Number[]  The offset of the area in which you are clustering. Clustering is performed for the visible area of the map. Use this option to expand the area of clustering relative to the visible area of the map.
<a href="#">options.zoomMargin</a>	0	Type: Number Number[]  Offset from the map viewport borders to observe when zooming in the map after a cluster is clicked. Recommended to set this option's value to match the size of the cluster icons and placemarks. For example, if only the bottom half of a placemark falls on the visible area of the map, a nonzero "top" offset should be set, so the placemark remains completely visible after the cluster is broken up. If a single number is set, it is applied to each side. If two numbers are set, they are the horizontal and vertical margins, respectively. If an array of four numbers is set, they are the top, right, bottom, and left margins.

**Example:**

```
// Creating a clusterer

// Creating a map that requires clusterization of geo objects
var map = new ymaps.Map('mapsID', { center: [56.034, 36.992], zoom: 8 });
// Creating an array of geo objects
myGeoObjects = [];
myGeoObjects[0] = new ymaps.GeoObject({
  geometry: { type: "Point", coordinates: [56.034, 36.992] },
  properties: {
    clusterCaption: 'Geo object №1',
    balloonContentBody: 'Balloon content for geo object №1'
  }
});
myGeoObjects[1] = new ymaps.GeoObject({
  geometry: { type: "Point", coordinates: [56.021, 36.983] },
  properties: {
    clusterCaption: 'Geo object №2',
    balloonContentBody: 'Balloon content for geo object №2'
  }
});

// Creating a cluster and prohibiting zooming the map in when the cluster is clicked
var clusterer = new ymaps.Clusterer({ clusterDisableClickZoom: true });
clusterer.add(myGeoObjects);
map.geoObjects.add(clusterer);
```

**Fields**

Name	Type	Description
<a href="#">balloon</a>	<a href="#">clusterer.Balloon</a>	Clusterer's balloon.
<a href="#">balloonclose</a>		Closing the balloon. <ul style="list-style-type: none"> <li>target - Link to the clusterer balloon manager.</li> <li>cluster - Reference to the cluster object.</li> </ul> Instance of the Event class.

Name	Type	Description
<a href="#">balloonopen</a>		Opening a balloon on a cluster. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>target - Link to the clusterer balloon manager.</li> <li>cluster - Reference to the cluster object.</li> </ul> Instance of the Event class.
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">hint</a>	clusterer.hint	Clusterer's hint.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">hintclose</a>	Closing the hint. Instance of the <a href="#">Event</a> class.
<a href="#">hintopen</a>	Opening a hint on a cluster. Instance of the <a href="#">Event</a> class.
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> Inherited from <a href="#">IParentOnMap</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .

## Methods

Name	Returns	Description
<a href="#">add(objects)</a>	<a href="#">Clusterer</a>	Adds a geo object or array of geo objects to the clusterer.

Name	Returns	Description
<a href="#">createCluster(center, geoObjects)</a>	<a href="#">IGeoObject</a>	Function for creating a cluster using the clusterer. Called directly by the clusterer during the clusterization process. Accepts as input the cluster center and an array of geo objects that go in this cluster. Returns the cluster, which will later be added to the map. If you need the clusterer to create user cluster objects, you should redefine this method for the clusterer.
<a href="#">getBounds()</a>	Number[][] null	Returns the geographical coordinates of the rectangular area that includes all the clusterer elements.
<a href="#">getClusters()</a>	<a href="#">IGeoObject[]</a>	Method for getting the current array of cluster objects. Note that the cluster objects change when you change the map zoom or shift its center. If you want to perform operations on all clusters, it is better to redefine the <a href="#">createCluster</a> method and add the required operations to its call.
<a href="#">getGeoObjects()</a>	<a href="#">IGeoObject[]</a>	Returns an array of geo objects added to the clusterer.
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getObjectState(geoObject)</a>	Object	Function for getting information about the current state of an object that was added to the clusterer.
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IChildOnMap</a> .
<a href="#">remove(objects)</a>	<a href="#">Clusterer</a>	Removes geo objects from the clusterer.
<a href="#">removeAll()</a>	<a href="#">Clusterer</a>	Removes all geo objects from the clusterer.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .

## Fields details

### balloon

```
{clusterer.Balloon} balloon
```

Clusterer's balloon.

### balloonclose

```
balloonclose
```

Closing the balloon.

- target - Link to the clusterer balloon manager.
- cluster - Reference to the cluster object.

Instance of the Event class.

### balloonopen

```
balloonopen
```

Opening a balloon on a cluster. Names of fields that are available via the [Event.get](#) method:

- target - Link to the clusterer balloon manager.
- cluster - Reference to the cluster object.

Instance of the Event class.

#### Example:

```
clusterer.events.add('balloonopen', function (e) {  
    var clusterPlacemark = e.get('cluster');  
});
```

### hint

```
{clusterer.hint} hint
```

Clusterer's hint.

## Events details

### hintclose

Closing the hint. Instance of the [Event](#) class.

### hintopen

Opening a hint on a cluster. Instance of the [Event](#) class.

## Methods details

### add

```
{Clusterer} add(objects)
```

Adds a geo object or array of geo objects to the clusterer.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>objects *</code>	—	Type: <code>IGeoObject GeoObject[]</code>  Array of geo objects, or a single geo object.

\* Mandatory parameter/option.

#### Example:

```
var myPlacemark = new ymaps.Placemark([35, 21]);
clusterer.add(myPlacemark);
var placemarks = [
  new ymaps.Placemark([44, 55]),
  new ymaps.Placemark([34, 45])
];
clusterer.add(placemarks);
```

### createCluster

```
{IGeoObject} createCluster(center, geoObjects)
```

Function for creating a cluster using the clusterer. Called directly by the clusterer during the clusterization process. Accepts as input the cluster center and an array of geo objects that go in this cluster. Returns the cluster, which will later be added to the map. If you need the clusterer to create user cluster objects, you should redefine this method for the clusterer.

**Returns** the cluster object. By default, creates instances of the [ClusterPlacemark](#).

#### Parameters:

Parameter	Default value	Description
<code>center *</code>	—	Type: <code>Number[]</code>  The cluster center in geocoordinates.
<code>geoObjects *</code>	—	Type: <code>IGeoObject[]</code>  Array of placemarks in the cluster.

\* Mandatory parameter/option.

#### Example:

```
// Setting a hint for clusters depending on their contents.
var clusterer = new ymaps.Clusterer();
clusterer.createCluster = function (center, geoObjects) {
  // Creating a cluster placemark using a standard implementation of the method.
  var clusterPlacemark = ymaps.Clusterer.prototype.createCluster.call(this, center, geoObjects),
    geoObjectsLength = clusterPlacemark.getGeoObjects().length,
    hintContent;
  if (geoObjectsLength < 10) {
    hintContent = "Very few placemarks";
  } else if (geoObjectsLength < 100) {
    hintContent = "That's OK with placemarks";
  } else {
    hintContent = "A lot of placemarks";
  }
  clusterPlacemark.properties.set("hintContent", hintContent);
  return clusterPlacemark;
};
```

### getBounds

```
{Number[][]|null} getBounds()
```

**Returns** the geographical coordinates of the rectangular area that includes all the clusterer elements.

**Example:**

```
var clusterer = new ymaps.Clusterer();
clusterer.add(myPlacemarks);
map.setBounds(clusterer.getBounds());
map.geoObjects.add(clusterer);
```

### getClusters

```
{IGeoObject[]} getClusters()
```

Method for getting the current array of cluster objects. Note that the cluster objects change when you change the map zoom or shift its center. If you want to perform operations on all clusters, it is better to redefine the createCluster method and add the required operations to its call.

**Returns** an array of clusters added to the map at this moment.

**Example:**

```
map.events.add('boundschange', function () {
    var clusters = clusterer.getClusters();
    alert('The map has moved and ' + clusters.length + ' clusters are displayed now.');
```

### getGeoObjects

```
{IGeoObject[]} getGeoObjects()
```

**Returns** an array of geo objects added to the clusterer.

**Example:**

```
// Counting objects in the visible area of the map.
var geoObjects = clusterer.getGeoObjects(),
    shownObjectsCounter = 0;
for (var i = 0, l = geoObjects.length; i < l; i++) {
    if (clusterer.getObjectState(geoObjects[i]).isShown) {
        shownObjectsCounter++;
    }
}
alert('shownObjectsCounter + ' out of ' + geoObjects.length + ' placemarks are displayed on the map.');
```

### getObjectState

```
{Object} getObjectState(geoObject)
```

Function for getting information about the current state of an object that was added to the clusterer.

**Returns** object with following fields:

- isShown - Indicates whether an object is in the visible area of the map.
- cluster - A reference to the cluster the object was added to.
- isClustered - Indicates whether the object was put in the cluster.

**Parameters:**

Parameter	Default value	Description
geoObject *	—	Type: <a href="#">IGeoObject</a>  The geo object to get the state for.

\* Mandatory parameter/option.

**Example:**

```
// Opening the cluster balloon with the selected object.
// Getting the info about the object's state within the cluster.
var geoObjectState = clusterer.getObjectState(myGeoObjects[1]);
// Checking if the object is located inside the visible area of the map.
if (geoObjectState.isShown) {
  // If the object is put in the cluster, the cluster balloon with the appropriate object is opened.
  if (geoObjectState.isClustered) {
    geoObjectState.cluster.state.set('activeObject', myGeoObjects[1]);
    clusterer.balloon.open(geoObjectState.cluster);
  } else {
    // If the object was not in the cluster, its own balloon is opened.
    myGeoObjects[1].balloon.open();
  }
}
```

**remove**

```
{Clusterer} remove(objects)
```

Removes geo objects from the clusterer.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>objects</code> *	—	Type: <a href="#">IGeoObject</a>   <a href="#">IGeoObject[]</a>  Array of geo objects.

\* Mandatory parameter/option.

**Example:**

```
var myPlacemark = new ymaps.Placemark([35, 21]),
    placemarks = [
      new ymaps.Placemark([44, 55]),
      new ymaps.Placemark([34, 45]),
      ...
    ];
clusterer.add(myPlacemark).add(placemarks);
...
// Deleting the first 10 objects of the array from the clusterer.
clusterer.remove(placemarks.slice(0, 10));
```

**removeAll**

```
{Clusterer} removeAll()
```

Removes all geo objects from the clusterer.

**Returns** self-reference.

**Example:**

```
clusterer.add(placemark1);
clusterer.add(placemark2);
clusterer.removeAll();
```

## ClusterPlacemark

Extends [IGeoObject](#), [collection.Item](#).

Cluster of geo objects. Used by default in [Clusterer](#).

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

## Constructor

```
ClusterPlacemark(geometry, properties[, options])
```

### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">Number[] Object IPointGeometry</a>  Coordinates of the placemark, or a hash describing the geometry, or a reference to the point geometry object.
<a href="#">properties</a> *	—	Type: <a href="#">IDataManager</a>  Cluster data.
<a href="#">properties.geoObjects</a> *	—	Type: <a href="#">IGeoObject[]</a>  Array of geo objects that are located in the given cluster.
<a href="#">options</a>	—	Type: <a href="#">Object</a>  Cluster options. In addition to special options, the cluster balloon supports the same options as <a href="#">Balloon</a> . The cluster balloon options are indicated with the "balloon" prefix.

Parameter	Default value	Description
<a href="#">options.balloonContentLayout</a>	'cluster#balloonTwoColumns'	<p>Type: Function String</p> <p>Layout of the cluster balloon in normal mode. You can pass the object's constructor with the <a href="#">ILayout</a> interface or the key for one of the standard layouts. Each standard layout has several customized options. In standard layouts, geo object data fields are used by default: <code>clusterCaption</code>, <code>balloonContentHeader</code>, <code>balloonContent</code>, and <code>balloonContentFooter</code> (see <a href="#">Placemark</a>).</p> <ul style="list-style-type: none"> <li>'cluster#balloonTwoColumns' - Two-column layout. The left column lists the names of placemarks (the <code>clusterCaption</code> field of the placemark or <code>balloonContentHeader</code>, if the first field is not defined). The right column contains information about the geo object. Layout options: <ul style="list-style-type: none"> <li><code>balloonLeftColumnWidth</code> — The width of the column with the list of objects in the cluster balloon, in pixels. Default value: 125</li> </ul> </li> <li>'cluster#balloonCarousel' - Information about the geo object is positioned in the center. The buttons to navigate to the previous and next geo object are positioned on each side. The lower part of the balloon has a quick navigation menu. Layout options: <ul style="list-style-type: none"> <li><code>balloonCycling</code> — Cycle the list when navigating with the side arrows. Default value: true</li> <li><code>balloonPagerSize</code> — Number of navigational items in the lower panel. The default value depends on the type of device. For mobile phones it is 4, and for tablets and PCs — 9.</li> <li><code>balloonPagerType</code> — Type of lower panel for navigation. Takes the values 'numeric' or 'marker'. <ul style="list-style-type: none"> <li>numeric — Display the number of a geo object in the list.</li> <li>marker — Display markers without numbers. Recommended for use when the number of items in the cluster is less than or equal to the value of <code>options.balloonPagerSize</code>. Default value: numeric</li> </ul> </li> <li><code>balloonPagerVisible</code> — Whether to display the navigation panel. Default value: true</li> </ul> </li> <li>'cluster#balloonAccordion' - Information about geo objects in list format. Each list item is a small icon and the name of the placemark (the <code>clusterCaption</code> field of the</li> </ul>

Parameter	Default value	Description
<a href="#">options.balloonContentLayoutHeight</a>	—	<p>Type: Number</p> <p>The height of the layout of the content of the balloon cluster. The balloon cluster option <code>balloonMaxHeight</code> is not set by default, because all standard layouts have fixed dimensions. The standard value depends on the layout.</p> <ul style="list-style-type: none"> <li>'cluster#balloonTwoColumns' - 210 pixels</li> <li>'cluster#balloonCarousel' - 177 pixels</li> <li>'cluster#balloonAccordion' - 283 pixels</li> </ul>
<a href="#">options.balloonContentLayoutWidth</a>	—	<p>Type: Number</p> <p>The width of the layout of the content of the balloon cluster. The balloon cluster option <code>balloonMaxWidth</code> is not set by default, because all standard layouts have fixed dimensions. The standard value depends on the layout.</p> <ul style="list-style-type: none"> <li>'cluster#balloonTwoColumns' - 475 pixels</li> <li>'cluster#balloonCarousel' - 308 pixels</li> <li>'cluster#balloonAccordion' - 305 pixels</li> </ul> <p>The option is not used in the panel mode.</p>
<a href="#">options.balloonItemContentLayout</a>	—	<p>Type: <code>ILayout</code> String</p> <p>The layout containing the information about geo object. The standard value depends on the layout.</p> <ul style="list-style-type: none"> <li>Displayed to the right of the list in the 'cluster#balloonTwoColumns' layout. The standard value of 'cluster#balloonTwoColumnsItemContent'</li> <li>Displayed in the center in the 'cluster#balloonCarousel' layout. The standard value of 'cluster#balloonCarouselItemContent'</li> <li>Displayed after the list item is clicked in the 'cluster#balloonAccordion' layout. The standard value of 'cluster#balloonAccordionItemContent'.</li> </ul> <p><b>The set of fields received by this layout differs from the parent and matches the fields that received by the standard layout of a geo object balloon. Additionally, <code>ownerProperties</code>, <code>ownerOptions</code> and <code>ownerState</code> fields have been added to access the cluster data.</b></p>

Parameter	Default value	Description
<a href="#">options.balloonPanelContentLayout</a>	null	Type: Function String  Layout of the cluster balloon in the "panel" mode. You can pass the object's constructor with the <a href="#">ILayout</a> interface. The available values are the same as in the 'balloonContentLayout' options. If null, then the value of the 'balloonContentLayout' option is used.
<a href="#">options.cursor</a>	'pointer'	Type: String  Cursor over the cluster marker.
<a href="#">options.disableClickZoom</a>	false	Type: Boolean  Flag that prohibits zooming in on the map when the cluster is clicked.
<a href="#">options.hideIconOnBalloonOpen</a>	true	Type: Boolean  Hide the icon when opening the balloon.
<a href="#">options.iconColor</a>	—	Type: String  The color of the cluster icon. This option is used for standard icons in browsers that support SVG.
<a href="#">options.iconContentLayout</a>	'cluster#iconContent'	Type: Function String  Layout for cluster marker contents. (Type: constructor for an object with the <a href="#">ILayout</a> interface or the layout key). If you do not want to display the content of the placemark, set the option value to null.
<a href="#">options.iconLayout</a>	'cluster#icon'	Type: Function String  Layout of the cluster placemark (Type: constructor for an object with the <a href="#">ILayout</a> interface or the layout key).

Parameter	Default value	Description
<a href="#">options.icons</a>	—	<p>Type: Object[]</p> <p>Array describing the icon for standard implementation of the cluster. An icon description is made up of an object with the following fields:</p> <ul style="list-style-type: none"> <li>href - Link to the image.</li> <li>size - Array of two numbers representing the icon size in pixels.</li> <li>offset - Offset of the icon relative to the anchor point of the object.</li> <li>shape - Optional field. An object that implements the <a href="#">IShape</a> interface or a JSON description of the geometry. Allows to specify the description of the icon geometry. If the parameter is omitted, a rectangular area around the icon will be considered as an active area for events (mouse over, mouse click).</li> </ul>
<a href="#">options.iconShape</a>	—	<p>Type: <a href="#">IGeometryJson</a> null</p> <p>The hotspot shape of the cluster. Specified as a JSON description of the pixel geometry of the icon. Use this option when creating your HTML layouts. The coordinates of the figure geometry are counted from the anchor point.</p>
<a href="#">options.interactivityModel</a>	'default#geoObject'	<p>Type: String</p> <p>Cluster interactivity model. Available keys and their values are listed in the description of <a href="#">interactivityModel.storage</a>.</p>
<a href="#">options.numbers</a>	[10, 100]	<p>Type: Number[]</p> <p>Array describing the boundary values for cluster dimensions. The number of icons described in the "icons" options must be 1 more than the number in this array.</p>
<a href="#">options.openBalloonOnClick</a>	true	<p>Type: Boolean</p> <p>Option that prevents opening the balloon when clicking on a cluster. By default, opening the balloon is allowed.</p>
<a href="#">options.openEmptyHint</a>	false	<p>Type: Boolean</p> <p>Enables displaying empty popup hints. Empty cluster hints are not shown by default.</p>

Parameter	Default value	Description
<a href="#">options.openHintOnHover</a>	true	Type: Boolean  Option that allows you to forbid displaying the popup hint when the cluster is pointed at. By default, showing hints is allowed.
<a href="#">options.zIndexHover</a>	—	Type: Number  The zIndex value, which is set to the cluster placemark on mouse hovering.

\* Mandatory parameter/option.

## Examples:

### 1.

```
// Variable with a description of two types of cluster icons.
var clusterIcons = [
  {
    href: 'small.png',
    size: [40, 40],
    // Offset so the image center matches the cluster center.
    offset: [-20, -20]
  },
  {
    href: 'big.png',
    size: [60, 60],
    offset: [-30, -30],
    // You can define the shape of the hotspot area
    // of the placemark.
    // The rectangular area is used by default.
    shape: {
      type: 'Circle',
      coordinates: [0, 0],
      radius: 30
    }
  }
],
// For a cluster size up to 100, the image small.jpg will be used.
// For a cluster size over 100, big.png will be used.
clusterNumbers = [100],
// Making a layout for cluster icon contents,
// in which numbers will be white.
MyIconContentLayout = ymaps.templateLayoutFactory.createClass(
  '<div style="color: #FFFFFF; font-weight: bold;">{{ properties.geoObjects.length }}</div>',
),
var clusterer = new ymaps.Clusterer({
  // If options for the cluster are set via the clusterer,
  // they must be specified with the "cluster" prefix.
  clusterIcons: clusterIcons,
  clusterNumbers: clusterNumbers,
  clusterIconContentLayout: MyIconContentLayout
});
```

### 2.

```
// Creating a clusterer with a carousel layout
var clusterer = new ymaps.Clusterer({
  clusterDisableClickZoom: true,
  // Using the "carousel" layout
  clusterBalloonContentLayout: "cluster#balloonCarousel",
  // Prohibiting cycling the list for paginated navigation
  clusterBalloonCycling: false,
  // Configuring the appearance of the navigation panel.
  // Markers will be the items in the navigation panel.
  clusterBalloonPagerType: "marker",
  // Number of items in the navigation panel
  clusterBalloonPagerSize: 6
});
```

### 3.

```
// Creating a clusterer with an accordian layout
var clusterer = new ymaps.Clusterer({
  clusterDisableClickZoom: true,
  // Using the "accordian" layout
  clusterBalloonContentLayout: "cluster#balloonAccordion"
```

```
});
```

#### 4.

```
// Creating a clusterer with an arbitrary HTML layout of the cluster icon.
var clusterer = new ymaps.Clusterer({
  // Defining the cluster placemark layout.
  clusterIconLayout: ymaps.templateLayoutFactory.createClass('<div
class="clusterIcon">{{ properties.geoObjects.length }}</div>'),
  // Redefining the active area of the placemark to make it clickable.
  clusterIconShape: {
    type: 'Rectangle',
    coordinates: [[0, 0], [20, 20]]
  }
});
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">IDataManager</a>	Geo object data. Inherited from <a href="#">IGeoObject</a> .
<a href="#">state</a>	<a href="#">data.Manager</a>	Cluster state. Defined by the following fields: <ul style="list-style-type: none"> <li><code>activeObject</code> - Reference to the cluster's active object. The active object is the one that is currently selected in the cluster balloon.</li> </ul>

### Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometrychange</a>	Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the geometry.</li> </ul> Inherited from <a href="#">IGeoObject</a> .

Name	Description
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldMap</code> - Old map.</li><li>• <code>newMap</code> - New map.</li></ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li><code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li><code>oldParent</code> - Old parent.</li> <li><code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getBounds()</a>	<code>Number[][] null</code>	Returns the geographical coordinates of the rectangular area that includes all the cluster elements.
<a href="#">getGeoObjects()</a>	<a href="#">IGeoObject[]</a>	The method is a simplified call of <code>cluster.properties.get('geoObjects')</code> ;
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .

Name	Returns	Description
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.  Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay.  Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IChildOnMap</a> .
<a href="#">onAddToMap(map)</a>		Function that is called when adding an element to the map. To perform additional actions when adding the object to the map, redefine this function.  Inherited from <a href="#">collection.Item</a> .
<a href="#">onRemoveFromMap(oldMap)</a>		Function that is called when deleting an element from the map. To perform additional actions when deleting the object from the map, redefine this function.  Inherited from <a href="#">collection.Item</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .

## Fields details

### state

```
{data.Manager} state
```

Cluster state. Defined by the following fields:

- `activeObject` - Reference to the cluster's active object. The active object is the one that is currently selected in the cluster balloon.

### Example:

```
var geoObjects = cluster.properties.get('geoObjects');
// When opening the cluster's balloon, the third object in the list will be selected.
cluster.state.set('activeObject', geoObjects[2]);
```

## Methods details

### getBounds

```
{Number[][]|null} getBounds()
```

Returns the geographical coordinates of the rectangular area that includes all the cluster elements.

### getGeoObjects

```
{IGeoObject[]} getGeoObjects()
```

The method is a simplified call of `cluster.properties.get('geoObjects')`;

Returns an array of geo objects that make up the cluster.

## collection

### collection.Item

Extends [IChildOnMap](#), [ICustomizable](#), [IEventEmitter](#), [IParentOnMap](#).

Base class for an item in a collection of map objects.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
collection.Item([options])
```

### Parameters:

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object Object options.

### Example:

```
// Example of implementing a custom control based on inheritance from collection.Item.
// The control displays the name of the object that is located in the center of the map.
var map = new ymaps.Map('map', {
  center: [55.819543, 37.611619],
  zoom: 6
});
// Creating a custom class.
var CustomControl = function (options) {
  CustomControl.superclass.constructor.call(this, options);
};
// And inheriting from collection.Item.
ymaps.util.defineClass(CustomControl, ymaps.collection.Item, {
  onAddToMap: function (map) {
    CustomControl.superclass.onAddToMap.call(this, map);
    // Creating an HTML element with text.
    this.getParent().getChildElement(this).then(this._onChildElementGet, this);
  },
  onRemoveFromMap: function (oldMap) {
    CustomControl.superclass.onRemoveFromMap.call(this, oldMap);
  },
  _onChildElementGet: function (parentElementContainer) {
    // You can create a DOM representation for the control here
    // and add it as a child element in parentElementContainer.
    // ...
  }
});
var customControl = new CustomControl();
map.controls.add(customControl, {top: 10, left: 10});
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

**Events**

Name	Description
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> Inherited from <a href="#">IParentOnMap</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .

**Methods**

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns the map that the collection item belongs to.
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>	Returns parent object.
<a href="#">onAddToMap(map)</a>		Function that is called when adding an element to the map. To perform additional actions when adding the object to the map, redefine this function.
<a href="#">onRemoveFromMap(oldMap)</a>		Function that is called when deleting an element from the map. To perform additional actions when deleting the object from the map, redefine this function.
<a href="#">setParent(parent)</a>	<a href="#">collection.Item</a>	Sets the parent for the selected item in a collection.

## Methods details

### getMap

```
{Map} getMap()
```

**Returns** the map that the collection item belongs to.

### getParent

```
{IParentOnMap} getParent()
```

**Returns** parent object.

### onAddToMap

```
{ } onAddToMap (map)
```

Function that is called when adding an element to the map. To perform additional actions when adding the object to the map, redefine this function.

#### Parameters:

Parameter	Default value	Description
<code>map</code> *	—	Type: <code>Map</code> The map that the object has been added to.

\* Mandatory parameter/option.

### onRemoveFromMap

```
{ } onRemoveFromMap (oldMap)
```

Function that is called when deleting an element from the map. To perform additional actions when deleting the object from the map, redefine this function.

#### Parameters:

Parameter	Default value	Description
<code>oldMap</code> *	—	Type: <code>Map</code> The map that the object has been deleted from.

\* Mandatory parameter/option.

### setParent

```
{collection.Item} setParent (parent)
```

Sets the parent for the selected item in a collection.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<a href="#">parent</a> *	—	Type: <a href="#">IParentOnMap</a> Parent object.

\* Mandatory parameter/option.

## Collection

Extends [ICollection](#), [collection.Item](#).

Basic implementation of an object collection on the map.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
Collection([options])
```

#### Parameters:

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object Collection options.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

### Events

Name	Description
<a href="#">add</a>	A child object was added. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>child - Child element that was added.</li> </ul>
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> Inherited from <a href="#">IParentOnMap</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .

Name	Description
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">remove</a>	<p>A child object was deleted. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>child - Child element that was deleted.</li> </ul>

## Methods

Name	Returns	Description
<a href="#">add(child)</a>	<a href="#">Collection</a>	Adds an item to the collection.
<a href="#">each(callback[, context])</a>	<a href="#">Collection</a>	Calls a handler function for all the items in the collection.
<a href="#">filter(filterFunction)</a>	Object[]	Calls a filter function for all the items in the collection. When the filter returns a non-zero value, the collection item goes in the final array.
<a href="#">get(index)</a>	Object	Returns a collection item, or null if the number is outside of the range for the collection's numbers.
<a href="#">getAll()</a>	Object[]	Returns an array with all the collection items.
<a href="#">getIterator()</a>	<a href="#">Iterator</a>	Returns an iterator for selecting collection items.
<a href="#">getLength()</a>	Number	Returns the number of items in the collection.
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns the map that the collection item belongs to. Inherited from <a href="#">collection.Item</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>	Returns parent object. Inherited from <a href="#">collection.Item</a> .
<a href="#">indexOf(childToFind)</a>	Number	Returns the sequential number of the object in the collection, or -1 if the object was not found.

Name	Returns	Description
<a href="#">onAddToMap(map)</a>		Function that is called when adding an element to the map. To perform additional actions when adding the object to the map, redefine this function.  Inherited from <a href="#">collection.Item</a> .
<a href="#">onRemoveFromMap(oldMap)</a>		Function that is called when deleting an element from the map. To perform additional actions when deleting the object from the map, redefine this function.  Inherited from <a href="#">collection.Item</a> .
<a href="#">remove(child)</a>	<a href="#">Collection</a>	Deletes an item from a collection.
<a href="#">removeAll()</a>	<a href="#">Collection</a>	Deletes all the items from a collection.
<a href="#">setParent(parent)</a>	<a href="#">collection.Item</a>	Sets the parent for the selected item in a collection.  Inherited from <a href="#">collection.Item</a> .

## Events details

### add

A child object was added. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- child - Child element that was added.

### remove

A child object was deleted. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- child - Child element that was deleted.

## Methods details

### add

```
{Collection} add(child)
```

Adds an item to the collection.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<a href="#">child</a> *	—	Type: <a href="#">collection.Item</a>  Element to add.

\* Mandatory parameter/option.

### each

```
{Collection} each(callback[, context])
```

Calls a handler function for all the items in the collection.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>callback</code> *	—	Type: Function  Handler function. Receives a collection item as input. If the function returns the value "false," processing stops.
<code>context</code>	—	Type: Object  Context of the invoked function.

\* Mandatory parameter/option.

### filter

```
{Object[]} filter(filterFunction)
```

Calls a filter function for all the items in the collection. When the filter returns a non-zero value, the collection item goes in the final array.

**Returns** an array of items that were selected.

#### Parameters:

Parameter	Default value	Description
<code>filterFunction</code> *	—	Type: Function  Function that works as a filter for the collection's objects. It takes an item from the collection as the first parameter. It should return a boolean value.

\* Mandatory parameter/option.

### get

```
{Object} get(index)
```

**Returns** a collection item, or null if the number is outside of the range for the collection's numbers.

#### Parameters:

Parameter	Default value	Description
<code>index *</code>	—	Type: Number  The sequential number of the item in the collection.

\* Mandatory parameter/option.

### getAll

```
{Object[]} getAll()
```

**Returns** an array with all the collection items.

### getIterator

```
{Iterator} getIterator()
```

**Returns** an iterator for selecting collection items.

### getLength

```
{Number} getLength()
```

**Returns** the number of items in the collection.

### indexOf

```
{Number} indexOf(childToFind)
```

**Returns** the sequential number of the object in the collection, or -1 if the object was not found.

#### Parameters:

Parameter	Default value	Description
<code>childToFind *</code>	—	Type: Object  The required object.

\* Mandatory parameter/option.

### remove

```
{Collection} remove(child)
```

Deletes an item from a collection.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>child *</code>	—	Type: <code>collection.Item</code>  Item to delete.

\* Mandatory parameter/option.

**removeAll**

```
{Collection} removeAll()
```

Deletes all the items from a collection.

**Returns** self-reference.

**control****control.Button**

Extends [ICustomizable](#), [ISelectableControl](#).

The "Button" control. The standard button layout changes its appearance depending on the size of the map. If the map is wide enough, the "image + text" version of the button is shown. If the map is medium in size, the "text" version of the button is shown. If the map is small, only the icon is shown in the button layout. If a button has no icon, then only text will be displayed in all states, and vice versa.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
control.Button([parameters])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">parameters</a>	—	Type: Object String Parameters of a button or string - button contents in HTML format.
<a href="#">parameters.data</a>	—	Type: Object Button data.
<a href="#">parameters.data.content</a>	—	Type: String Button contents in HTML format.
<a href="#">parameters.data.image</a>	—	Type: String URL of the button icon. The standard layout of a button is designed for the icon size 16x16 pixels.
<a href="#">parameters.data.title</a>	—	Type: String Text of the popup hint that appears when the mouse cursor hovers over the button.
<a href="#">parameters.options</a>	—	Type: Object Button options.
<a href="#">parameters.options.adjustMapMargin</a>	false	Type: Boolean Whether the button registers its size in the map margins manager <a href="#">map.margin.Manager</a> .

Parameter	Default value	Description
<a href="#">parameters.options.float</a>	"right"	<p>Type: String</p> <p>The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map. See also the description of the position option.</p>
<a href="#">parameters.options.floatIndex</a>	0	<p>Type: Number</p> <p>The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none".</p>
<a href="#">parameters.options.layout</a>	—	<p>Type: <a href="#">ISelectableControlLayout</a> String</p> <p>Constructor of the control layout which implements the <a href="#">ISelectableControlLayout</a> interface or the layout key in the <a href="#">layout.storage</a>. The layout constructor is passed an object containing the fields:</p> <ul style="list-style-type: none"> <li>control - Reference to the control.</li> <li>options - Control options manager <a href="#">control.Button.options</a>.</li> <li>data - Control data manager <a href="#">control.Button.data</a>.</li> <li>state - Control state manager <a href="#">control.Button.state</a>.</li> </ul> <p>The layout's outward appearance changes based on the control's data, state and options. The control, in turn, reacts to layout interface events and changes the values of fields for <a href="#">control.Button.state</a> depending on the commands received.</p>
<a href="#">parameters.options.maxWidth</a>	90	<p>Type: Number Number[]</p> <p>The maximum width of the button in different states. If a number is specified, it is assumed that the button has the same maximum dimensions in all states. If an array is specified, it will be interpreted as the maximum width of the button in different states, from the lesser to the greater. The number of states is set in the instance of the class <a href="#">control.Manager</a>, which is usually a field of <a href="#">Map.controls</a>, via the "states" option. By default, the controls have three states ['small', 'medium', 'large'].</p>

Parameter	Default value	Description
<a href="#">parameters.options.position</a>	—	Type: Object  Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".
<a href="#">parameters.options.position.bottom</a>	'auto'	Type: Number String  Position relative to the bottom edge of the map.
<a href="#">parameters.options.position.left</a>	'auto'	Type: Number String  Position relative to the left edge of the map.
<a href="#">parameters.options.position.right</a>	'auto'	Type: Number String  Position relative to the right edge of the map.
<a href="#">parameters.options.position.top</a>	'auto'	Type: Number String  Position relative to the top edge of the map.
<a href="#">parameters.options.selectOnClick</a>	true	Type: Boolean  Options describing button behavior. <ul style="list-style-type: none"> <li>If true, the button becomes "pressed" after the click and changes its appearance and the value of the <a href="#">control.Button.state</a> field is changed to "selected".</li> <li>false - The button's appearance and state does not change after clicking it.</li> </ul>
<a href="#">parameters.options.size</a>	'auto'	Type: String  Defines the appearance of the standard button layout. Takes the following values: <ul style="list-style-type: none"> <li>'auto' - The default button layout is changed automatically depending on the dimensions of the map and the number of added controls.</li> <li>'small' - The button layout displays the icon, regardless of the map size.</li> <li>'medium' - The button layout displays only text, regardless of the map size.</li> <li>'large' - The button layout always displays both the icon and text, regardless of the map size.</li> </ul>
<a href="#">parameters.options.visible</a>	true	Type: Boolean  Indicates if the control is displayed.

Parameter	Default value	Description
<code>parameters.state</code>	—	Type: Object  Object describing the state of the button.
<code>parameters.state.enabled</code>	true	Type: Boolean  Flags if the button is active.
<code>parameters.state.selected</code>	false	Type: Boolean  Flags if the button is pressed.

**Examples:****1.**

```
// Example 1.
// Creating a button and adding it to the map.
var button = new ymaps.control.Button({
  data: {
    // Setting an icon for the button.
    image: 'images/button.jpg',
    // Text on the icon.
    content: 'Save',
    // Text for the popup hint.
    title: 'Click to save the route'
  },
  options: {
    // Setting up the button options.
    selectOnClick: false,
    // The button will have three states - icon, text, and icon + text.
    // So we'll define three values for the button width for all states.
    maxWidth: [30, 100, 150]
  }
});
map.controls.add(button, { float: 'right', floatIndex: 100 });
```

**2.**

```
// Example 2.
// Creating buttons with a custom layout
var button = new ymaps.control.Button({
  data: {
    content: 'Red button',
    title: 'Press the button'
  },
  options: {
    layout: ymaps.templateLayoutFactory.createClass(
      // If the button is not pressed, the css style 'myButton' will be applied
      // If the button is pressed, the css style 'myButton' and 'myButtonSelected' will be applied.
      "<div class='myButton {% if state.selected %}myButtonSelected{% endif %}' title='{{ data.title }}'>" +
      "{{ data.content }}" +
      "</div>"
    ),
    // In order to correctly position all other controls horizontally,
    // you should specify the maximum width of the layout.
    maxWidth: 150
  }
});
map.controls.add(button, { float: 'left', floatIndex: 0 });

// You can define the positioning relative to the edges of the map. In this case,
// the value of the maxWidth option does not affect
// the positioning of controls.
map.controls.add(button, { float: 'none', position: {left: '5px', top: '5px'} });
```

**Fields**

Name	Type	Description
<a href="#">data</a>	<a href="#">data.Manager</a>	<p>Button data. Names of fields that are available via the <code>data.Manager.get</code> method:</p> <ul style="list-style-type: none"> <li><code>image</code> - Button icon, if available.</li> <li><code>content</code> - Button content in HTML format.</li> <li><code>title</code> - Text of the popup hint that appears when the mouse cursor hovers over the button.</li> </ul>
<a href="#">events</a>	<a href="#">IEventManager</a>	<p>Event manager.</p> <p>Inherited from <a href="#">IEventEmitter</a>.</p>
<a href="#">options</a>	<a href="#">IOptionManager</a>	<p>Options manager.</p> <p>Inherited from <a href="#">IControl</a>.</p>
<a href="#">press</a>		<p>The event indicating that the button has been pressed. Unlike the click event, it is generated only if the state <code>isEnabled == true</code>. Instance of the Event class.</p>
<a href="#">state</a>	<a href="#">data.Manager</a>	<p>Button state. Names of fields that are available via the <code>data.Manager.get</code> method:</p> <ul style="list-style-type: none"> <li><code>selected</code> - Flag for whether the button is selected.</li> <li><code>enabled</code> - Flag for whether the button is active.</li> <li><code>size</code> - The size that is currently set for the button.</li> </ul>

**Events**

Name	Description
<a href="#">click</a>	Clicking the button. Instance of the <a href="#">Event</a> class.
<a href="#">deselect</a>	<p>The control is not selected.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">disable</a>	<p>The control is unavailable.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">enable</a>	<p>The control is available.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li><code>oldParent</code> - Old parent.</li> <li><code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">select</a>	<p>The control is selected.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">deselect()</a>		<p>Cancels selection of the control (turns it off).</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">disable()</a>		<p>Makes the control unavailable (user actions are not allowed).</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">enable()</a>		<p>Makes the control available (user actions are allowed).</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map.
<a href="#">getParent()</a>	<a href="#">IControlParent</a>  null	<p>Returns link to the parent object, or null if the parent element was not set.</p> <p>Inherited from <a href="#">IControl</a>.</p>
<a href="#">isEnabled()</a>	Boolean	<p>Returns true if the control is available, or false if it is unavailable.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">isSelected()</a>	Boolean	<p>Returns true if the control is selected, or false if it is not selected.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">select()</a>		<p>Selects (turns on) the control.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	<p>Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.</p> <p>Inherited from <a href="#">IControl</a>.</p>

**Fields details****data**

```
{data.Manager} data
```

Button data. Names of fields that are available via the `data.Manager.get` method:

- `image` - Button icon, if available.
- `content` - Button content in HTML format.

- `title` - Text of the popup hint that appears when the mouse cursor hovers over the button.

### press

```
press
```

The event indicating that the button has been pressed. Unlike the click event, it is generated only if the state `isEnabled == true`. Instance of the `Event` class.

### state

```
{data.Manager} state
```

Button state. Names of fields that are available via the `data.Manager.get` method:

- `selected` - Flag for whether the button is selected.
- `enabled` - Flag for whether the button is active.
- `size` - The size that is currently set for the button.

### Example:

```
var button = new ymaps.control.Button('Edit');
// Setting the button state to "selected" -
// similar to calling button.select();
button.state.set('selected', true);
```

### Events details

#### click

Clicking the button. Instance of the `Event` class.

### Methods details

#### getMap

```
{Map} getMap()
```

Returns reference to the map.

## control.FullscreenControl

Extends `control.Button`.

The "Fullscreen mode" control. You can set the z-index property of the map container for full-screen mode using the `Map.options.fullscreenZIndex` option. Key for the control in `control.storage` — "fullscreenControl".

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
control.FullscreenControl([parameters])
```

### Parameters:

Parameter	Default value	Description
<a href="#">parameters</a>	—	Type: Object Control parameters.

Parameter	Default value	Description
<a href="#">parameters.data</a>	—	Type: Object  Object describing the data of a control.
<a href="#">parameters.data.title</a>	—	Type: String  Text of the popup hint that appears when the mouse cursor hovers over the button.
<a href="#">parameters.options</a>	—	Type: Object  Control options.
<a href="#">parameters.options.adjustMapMargin</a>	false	Type: Boolean  Whether the control registers its size in the map margins manager <a href="#">map.margin.Manager</a> .
<a href="#">parameters.options.float</a>	"right"	Type: String  The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map.
<a href="#">parameters.options.floatIndex</a>	300	Type: Number  The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none". See also the description of the position option.

Parameter	Default value	Description
<a href="#">parameters.options.layout</a>	—	<p>Type: <a href="#">ISelectableControlLayout</a> String</p> <p>Constructor of the control layout which implements the <a href="#">ISelectableControlLayout</a> interface or the layout key in the <a href="#">layout.storage</a>. The layout constructor is passed an object containing the fields:</p> <ul style="list-style-type: none"> <li>control - Reference to the control.</li> <li>options - Options manager for the control.<a href="#">FullscreenControl.options</a>.</li> <li>data - Data manager for the control.<a href="#">FullscreenControl.data</a> control.</li> <li>state - Control state manager <a href="#">control.FullscreenControl.state</a>.</li> </ul> <p>The layout's outward appearance changes based on the control's data, state and options. The manager element, in turn, reacts to the layout's interface events.</p>
<a href="#">parameters.options.maxWidth</a>	28	<p>Type: Number Number[]</p> <p>The maximum width of the control in different states. If a number is specified, it is assumed that the control has the same maximum dimensions in all states. If an array is specified, it will be interpreted as the maximum width in different states from the lesser to the greater. The number of states is set in the instance of the class <a href="#">control.Manager</a>, which is usually a field of <a href="#">Map.controls</a>, via the "states" option. By default, the controls have three states ['small', 'medium', 'large']. By default, the control does not change its size and always looks like a button with an icon.</p>
<a href="#">parameters.options.position</a>	—	<p>Type: Object</p> <p>Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".</p>
<a href="#">parameters.options.position.bottom</a>	'auto'	<p>Type: Number String</p> <p>Position relative to the bottom edge of the map.</p>
<a href="#">parameters.options.position.left</a>	'auto'	<p>Type: Number String</p> <p>Position relative to the left edge of the map.</p>

Parameter	Default value	Description
<a href="#">parameters.options.position.right</a>	'auto'	Type: Number String Position relative to the right edge of the map.
<a href="#">parameters.options.position.top</a>	'auto'	Type: Number String Position relative to the top edge of the map.
<a href="#">parameters.options.visible</a>	true	Type: Boolean Indicates if the control is displayed.
<a href="#">parameters.state</a>	—	Type: Object Object describing the state of a control.
<a href="#">parameters.state.enabled</a>	true	Type: Boolean Flags if the button is active.
<a href="#">parameters.state.selected</a>	false	Type: Boolean Flags if the button is pressed.

**Example:**

```
// Adding the control to the map and immediately switching it
// to the full-screen mode.
var fullscreenControl = new ymaps.control.FullscreenControl();
myMap.controls.add(fullscreenControl);
fullscreenControl.enterFullscreen();
```

**Fields**

Name	Type	Description
<a href="#">data</a>	<a href="#">data.Manager</a>	Button data. Names of fields that are available via the <a href="#">data.Manager.get</a> method: <ul style="list-style-type: none"> <li>image - Button icon, if available.</li> <li>content - Button content in HTML format.</li> <li>title - Text of the popup hint that appears when the mouse cursor hovers over the button.</li> </ul> Inherited from <a href="#">control.Button</a> .
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">IControl</a> .
<a href="#">press</a>		The event indicating that the button has been pressed. Unlike the click event, it is generated only if the state <code>isEnabled == true</code> . Instance of the Event class. Inherited from <a href="#">control.Button</a> .

Name	Type	Description
<a href="#">state</a>	<a href="#">data.Manager</a>	State of the control. Names of fields that are available via the <code>data.Manager.get</code> method: <ul style="list-style-type: none"> <li><code>fullscreen</code> — Flag for whether the map is in full-screen mode.</li> </ul>

## Events

Name	Description
<a href="#">click</a>	Clicking the button. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">control.Button</a> .
<a href="#">deselect</a>	The control is not selected. Inherited from <a href="#">ISelectableControl</a> .
<a href="#">disable</a>	The control is unavailable. Inherited from <a href="#">ISelectableControl</a> .
<a href="#">enable</a>	The control is available. Inherited from <a href="#">ISelectableControl</a> .
<a href="#">fullscreenenter</a>	The map switched to fullscreen mode. Instance of the <a href="#">Event</a> class.
<a href="#">fullscreenexit</a>	The map exited full-screen mode. Instance of the <a href="#">Event</a> class.
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li><code>oldParent</code> - Old parent.</li> <li><code>newParent</code> - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .
<a href="#">select</a>	The control is selected. Inherited from <a href="#">ISelectableControl</a> .

## Methods

Name	Returns	Description
<a href="#">deselect()</a>		Cancels selection of the control (turns it off).  Inherited from <a href="#">ISelectableControl</a> .
<a href="#">disable()</a>		Makes the control unavailable (user actions are not allowed).  Inherited from <a href="#">ISelectableControl</a> .

Name	Returns	Description
<a href="#">enable()</a>		Makes the control available (user actions are allowed). Inherited from <a href="#">ISelectableControl</a> .
<a href="#">enterFullscreen()</a>		Allows you to switch the map to full-screen mode.
<a href="#">exitFullscreen()</a>		Allows you to take the map out of full-screen mode.
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">control.Button</a> .
<a href="#">getParent()</a>	<a href="#">IControlParent</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IControl</a> .
<a href="#">isEnabled()</a>	Boolean	Returns true if the control is available, or false if it is unavailable. Inherited from <a href="#">ISelectableControl</a> .
<a href="#">isSelected()</a>	Boolean	Returns true if the control is selected, or false if it is not selected. Inherited from <a href="#">ISelectableControl</a> .
<a href="#">select()</a>		Selects (turns on) the control. Inherited from <a href="#">ISelectableControl</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IControl</a> .

## Fields details

### state

```
{data.Manager} state
```

State of the control. Names of fields that are available via the `data.Manager.get` method:

- `fullscreen` — Flag for whether the map is in full-screen mode.

### Events details

#### fullscreenenter

The map switched to fullscreen mode. Instance of the [Event](#) class.

**fullscreenexit**

The map exited full-screen mode. Instance of the [Event](#) class.

**Methods details****enterFullscreen**

```
{ } enterFullscreen()
```

Allows you to switch the map to full-screen mode.

**exitFullscreen**

```
{ } exitFullscreen()
```

Allows you to take the map out of full-screen mode.

**control.GeolocationControl**

Extends [control.Button](#).

The "geolocation" control. Allows you to render the user's position on the map. The key of the control in the storage. [control.storage](#) — "geolocationControl".

See [geolocation](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
control.GeolocationControl([parameters])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">parameters</a>	—	Type: Object Control parameters.
<a href="#">parameters.data</a>	—	Type: Object Object describing the data of a control.
<a href="#">parameters.data.image</a>	'geolocation'	Type: String URL of the button icon.
<a href="#">parameters.data.title</a>	—	Type: String Text of the popup hint that appears when the mouse cursor hovers over the button.
<a href="#">parameters.options</a>	—	Type: Object Control options.
<a href="#">parameters.options.adjustMapMargin</a>	false	Type: Boolean Whether the control registers its size in the map margins manager <a href="#">map.margins.Manager</a> .

Parameter	Default value	Description
<a href="#">parameters.options.float</a>	"right"	Type: String  The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map. See also the description of the position option.
<a href="#">parameters.options.floatIndex</a>	300	Type: Number  The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none".
<a href="#">parameters.options.maxWidth</a>	28	Type: Number Number[]  The maximum width of the control in different states. If a number is specified, it is assumed that the control has the same maximum dimensions in all states. If an array is specified, it will be interpreted as the maximum width in different states from the lesser to the greater. The number of states is set in the instance of the class <a href="#">control.Manager</a> , which is usually a field of <a href="#">Map.controls</a> , via the "states" option. By default, the control does not change its size and always looks like a button with an icon.
<a href="#">parameters.options.noPlacemark</a>	false	Type: Boolean  When set to true, the location marker is not shown on the map, and automatic shifting of the map center and scaling does not occur.
<a href="#">parameters.options.position</a>	—	Type: Object  Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".
<a href="#">parameters.options.position.bottom</a>	'auto'	Type: Number String  Position relative to the bottom edge of the map.
<a href="#">parameters.options.position.left</a>	'auto'	Type: Number String  Position relative to the left edge of the map.

Parameter	Default value	Description
<a href="#">parameters.options.position.right</a>	'auto'	Type: Number String Position relative to the right edge of the map.
<a href="#">parameters.options.position.top</a>	'auto'	Type: Number String Position relative to the top edge of the map.
<a href="#">parameters.options.visible</a>	true	Type: Boolean Indicates if the control is displayed.
<a href="#">parameters.state</a>	—	Type: Object Object describing the state of a control.
<a href="#">options.useMapMargin</a>	true	Type: Boolean Whether to account for map margins <a href="#">map.margin.Manager</a> when centering the map.

**Example:**

```
// Adding the control with a custom geolocation placemark on the map.
var geolocationControl = new ymaps.control.GeolocationControl({
  options: {noPlacemark: true}
});
geolocationControl.events.add('locationchange', function (event) {
  var position = event.get('position');
  // When creating a placemark, you can set any appearance for it.
  var locationPlacemark = new ymaps.Placemark(position);

  myMap.geoObjects.add(locationPlacemark);
  // Setting the new map center to the user's current location.
  myMap.panTo(position);
});
myMap.controls.add(geolocationControl);
```

**Fields**

Name	Type	Description
<a href="#">data</a>	<a href="#">data.Manager</a>	Button data. Names of fields that are available via the <a href="#">data.Manager.get</a> method: <ul style="list-style-type: none"> <li>image - Button icon, if available.</li> <li>content - Button content in HTML format.</li> <li>title - Text of the popup hint that appears when the mouse cursor hovers over the button.</li> </ul> Inherited from <a href="#">control.Button</a> .
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">IControl</a> .
<a href="#">press</a>		The event indicating that the button has been pressed. Unlike the click event, it is generated only if the state <code>isEnabled == true</code> . Instance of the Event class. Inherited from <a href="#">control.Button</a> .

Name	Type	Description
<a href="#">state</a>	<a href="#">data.Manager</a>	<p>Button state. Names of fields that are available via the <a href="#">data.Manager.get</a> method:</p> <ul style="list-style-type: none"> <li>selected - Flag for whether the button is selected.</li> <li>enabled - Flag for whether the button is active.</li> <li>size - The size that is currently set for the button.</li> </ul> <p>Inherited from <a href="#">control.Button</a>.</p>

## Events

Name	Description
<a href="#">click</a>	<p>Clicking the button. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">control.Button</a>.</p>
<a href="#">deselect</a>	<p>The control is not selected.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">disable</a>	<p>The control is unavailable.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">enable</a>	<p>The control is available.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">locationchange</a>	<p>The event of determining the user's position. The list of event fields that are available via the <a href="#">Event.get</a>:</p> <ul style="list-style-type: none"> <li>position - The user's position, in geo coordinates.</li> <li>geoObjects — An instance of the <a href="#">GeoObjectCollection</a> class with the object that represents the user's current location.</li> </ul>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">select</a>	<p>The control is selected.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>

## Methods

Name	Returns	Description
<a href="#">deselect()</a>		<p>Cancels selection of the control (turns it off).</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>

Name	Returns	Description
<a href="#">disable()</a>		Makes the control unavailable (user actions are not allowed).  Inherited from <a href="#">ISelectableControl</a> .
<a href="#">enable()</a>		Makes the control available (user actions are allowed).  Inherited from <a href="#">ISelectableControl</a> .
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map.  Inherited from <a href="#">control.Button</a> .
<a href="#">getParent()</a>	<a href="#">IControlParent</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IControl</a> .
<a href="#">isEnabled()</a>	Boolean	Returns true if the control is available, or false if it is unavailable.  Inherited from <a href="#">ISelectableControl</a> .
<a href="#">isSelected()</a>	Boolean	Returns true if the control is selected, or false if it is not selected.  Inherited from <a href="#">ISelectableControl</a> .
<a href="#">select()</a>		Selects (turns on) the control.  Inherited from <a href="#">ISelectableControl</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IControl</a> .

### Events details

#### locationchange

The event of determining the user's position. The list of event fields that are available via the [Event.get](#):

- position - The user's position, in geo coordinates.
- geoObjects — An instance of the [GeoObjectCollection](#) class with the object that represents the user's current location.

### control.ListBox

Extends [ICollection](#), [IControl](#), [ICustomizable](#).

Class for creating a control in the form of a drop-down list. The standard drop-down list layout changes its appearance depending on the size of the map. If the map is wide enough, text can be displayed in the header of

a drop-down list. If the map is small in size, only an icon is displayed. If a button has no icon, then only text will be displayed in all states, and vice versa.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
control.ListBox([parameters])
```

### Parameters:

Parameter	Default value	Description
<a href="#">parameters</a>	—	Type: Object Drop-down list parameters.
<a href="#">parameters.data</a>	—	Type: Object Data.
<a href="#">parameters.data.content</a>	—	Type: String List title.
<a href="#">parameters.data.image</a>	—	Type: String URL of the button icon. The standard layout of a button is designed for the icon size 16x16 pixels.
<a href="#">parameters.data.title</a>	—	Type: String Text of the popup hint that appears when the mouse cursor hovers over the list.
<a href="#">parameters.items</a>	—	Type: <a href="#">IControl[]</a> Array of child elements of the list.
<a href="#">parameters.options</a>	—	Type: Object Control options.
<a href="#">parameters.options.adjustMapMargin</a>	false	Type: Boolean Whether the control registers its size in the map margins manager <a href="#">map.margin.Manager</a> .
<a href="#">parameters.options.collapseOnBlur</a>	true	Type: Boolean This flag enables to collapse the list when the button loses focus, such as when a user clicks on the document.
<a href="#">parameters.options.expandOnClick</a>	true	Type: Boolean Flag that allows automatically expanding/collapsing the list when clicked.

Parameter	Default value	Description
<a href="#">parameters.options.float</a>	"right"	<p>Type: String</p> <p>The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map. See also the description of the position option.</p>
<a href="#">parameters.options.floatIndex</a>	0	<p>Type: Number</p> <p>The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none".</p>
<a href="#">parameters.options.layout</a>	—	<p>Type: Function String</p> <p>Constructor of the control layout which implements the <a href="#">ISelectableControlLayout</a> and <a href="#">IGroupControlLayout</a> interfaces or the layout key in the <a href="#">layout.storage</a>. The layout constructor is passed an object containing the fields:</p> <ul style="list-style-type: none"> <li>control - Reference to the control.</li> <li>options - Control options manager <a href="#">control.ListBox.options</a>.</li> <li>data - Control data manager <a href="#">control.ListBox.data</a>.</li> <li>state - Control state manager <a href="#">control.ListBox.state</a>.</li> </ul> <p>The layout's outward appearance changes based on the control's data, state and options. The control, in turn, reacts to layout interface events and changes the values of fields for <a href="#">control.ListBox.state</a> depending on the commands received.</p>
<a href="#">parameters.options.maxWidth</a>	90	<p>Type: Number Number[]</p> <p>The maximum width of the listbox in different states. If a number is specified, it is assumed that the button has the same maximum dimensions in all states. If an array is specified, it will be interpreted as the maximum width of the button in different states, from the lesser to the greater. The number of states is set in the instance of the class <a href="#">control.Manager</a>, which is usually a field of <a href="#">Map.controls</a>, via the "states" option. By default, the controls have three states ['small', 'medium', 'large'].</p>

Parameter	Default value	Description
<a href="#">parameters.options.popupFloat</a>	—	Type: String  The side to align the manager element's popup to. Takes two values: "left" and "right". By default, it is defined by the "float" option.
<a href="#">parameters.options.position</a>	—	Type: Object  Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".
<a href="#">parameters.options.position.bottom</a>	'auto'	Type: Number String  Position relative to the bottom edge of the map.
<a href="#">parameters.options.position.left</a>	'auto'	Type: Number String  Position relative to the left edge of the map.
<a href="#">parameters.options.position.right</a>	'auto'	Type: Number String  Position relative to the right edge of the map.
<a href="#">parameters.options.position.top</a>	'auto'	Type: Number String  Position relative to the top edge of the map.
<a href="#">parameters.options.visible</a>	true	Type: Boolean  Indicates if the control is displayed.
<a href="#">parameters.state</a>	—	Type: Object  State of the drop-down list.
<a href="#">parameters.state.expanded</a>	false	Type: Boolean  Flags if the list is expanded.

**Examples:****1.**

```
// Example 1.
// Handling a click on list items.
var cityList = new ymaps.control.ListBox({
  data: {
    content: 'Select a city'
  },
  items: [
    new ymaps.control.ListBoxItem('Moscow'),
    new ymaps.control.ListBoxItem('Novosibirsk'),
    new ymaps.control.ListBoxItem({options: {type: 'separator'}}),
    new ymaps.control.ListBoxItem('New York'),
  ]
});
cityList.get(0).events.add('click', function () {
  map.setCenter([55.752736, 37.606815]);
});
cityList.get(1).events.add('click', function () {
  map.setCenter([55.026366, 82.907803]);
});
```

```
});
cityList.get(3).events.add('click', function () {
    map.setCenter([40.695537, -73.97552]);
});
map.controls.add(cityList, { floatIndex: 0 });
```

## 2.

```
// Example 2.
// Creating a custom list.
// This example uses jQuery, downloaded from http://yandex.st/jquery/1.6.4/jquery.min.js

// By default, the drop-down list reacts to the "click" event and automatically
// changes its state to expanded or collapsed.
var MyListBoxLayout = ymaps.templateLayoutFactory.createClass(
    '<div id="my-listbox-header" >{{ data.title }}</div >' +
    // This item will serve as a container for the child list items.
    '<div id="my-list-box" style="display: {% if state.expanded %}block{% else %}none{% endif %};" >' +
    '</div >', {

    build: function() {
        MyListBoxLayout.superclass.build.call(this);
        this.childContainerElement = $('#my-list-box').get(0);
        // Each time we rebuild, we will generate an event
        // that signals that the container for child elements has changed.
        // The event format is described in the IGroupControlLayout interface.
        this.events.fire('childcontainerchange', {
            newChildContainerElement: this.childContainerElement,
            oldChildContainerElement: null
        });
    },

    // Redefining the method that requires the IGroupControlLayout interface.
    getChildContainerElement: function () {
        return this.childContainerElement;
    }
});
// Creating a list and displaying the created layout via the options.
var listBox = new ymaps.control.ListBox({options: {layout: MyListBoxLayout}});
```

## 3.

```
// Example 3.
// In this example, the ListBox control filters objects to show
// on the map (multi-select is supported).
// ObjectManager is used here for adding objects to the map.

// Creating a drop-down list with 5 items.
var listBoxItems = ['School', 'Pharmacy', 'Store', 'Hospital', 'Bar']
    .map(function(title) {
        return new ymaps.control.ListBoxItem({
            data: {
                content: title
            },
            state: {
                selected: true
            }
        });
    });

// Now creating the drop-down list with 5 items.
var listBoxControl = new ymaps.control.ListBox({
    data: {
        content: 'Filter',
        title: 'Filter'
    },
    items: listBoxItems,
    state: {
        // Indicates that the list is expanded.
        expanded: true,
        filters: listBoxItems.reduce(function(filters, filter) {
            filters[filter.data.get('content')] = filter.isSelected();
            return filters;
        }, {})
    }
});

map.controls.add(listBoxControl);

// Adding tracking to the indicator to check if a list item is selected.
listBoxControl.events.add(['select', 'deselect'], function(e) {
    var listBoxItem = e.get('target');
    var filters = ymaps.util.extend({}, listBoxControl.state.get('filters'));
    filters[listBoxItem.data.get('content')] = listBoxItem.isSelected();
    listBoxControl.state.set('filters', filters);
});

// Tracking changes to the control.ListBox.state field.
var filterMonitor = new ymaps.Monitor(listBoxControl.state);
```

```

filterMonitor.add('filters', function(filters) {
    // Applying the filter to ObjectManager.
    objectManager.setFilter(getFilterFunction(filters));
});

function getFilterFunction(categories){
    return function(obj){
        var content = obj.properties.balloonContent;
        return categories[content]
    }
}

```

## Fields

Name	Type	Description
<a href="#">data</a>	<a href="#">data.Manager</a>	Drop-down list data. Names of fields that are available via the <a href="#">data.Manager.get</a> method: <ul style="list-style-type: none"> <li><code>content</code> - Title of the drop-down list.</li> <li><code>title</code> - Text of the popup hint that appears when the mouse cursor hovers over the list.</li> </ul>
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">IControl</a> .
<a href="#">state</a>	<a href="#">data.Manager</a>	State of the drop-down list. Names of fields that are available via the <a href="#">data.Manager.get</a> method: <ul style="list-style-type: none"> <li><code>expanded</code> - Flag for whether the list is expanded.</li> <li><code>size</code> - The size that is currently set for the list.</li> </ul>

## Events

Name	Description
<a href="#">add</a>	A child object was added. Inherited from <a href="#">ICollection</a> .
<a href="#">click</a>	Clicking the list title. Instance of the <a href="#">Event</a> class.
<a href="#">collapse</a>	The list is closed. Instance of the <a href="#">Event</a> class.
<a href="#">expand</a>	The list is open. Instance of the <a href="#">Event</a> class.
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li><code>oldParent</code> - Old parent.</li> <li><code>newParent</code> - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .
<a href="#">press</a>	The event indicating that the button has been pressed. Unlike the click event, it is generated only if the state <code>isEnabled == true</code> . Instance of the <a href="#">Event</a> class.
<a href="#">remove</a>	A child object was deleted. Inherited from <a href="#">ICollection</a> .

**Methods**

Name	Returns	Description
<a href="#">add(object)</a>	<a href="#">ICollection</a>	Adds a child object to the collection.  Inherited from <a href="#">ICollection</a> .
<a href="#">collapse()</a>	<a href="#">control.ListBox</a>	Collapses the list.
<a href="#">expand()</a>	<a href="#">control.ListBox</a>	Expands the list.
<a href="#">getIterator()</a>	<a href="#">IIterator</a>	Returns iterator for the collection.  Inherited from <a href="#">ICollection</a> .
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map.
<a href="#">getParent()</a>	<a href="#">IControlParent</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IControl</a> .
<a href="#">isExpanded()</a>	Boolean	Returns a flag for whether the control is in the expanded state.
<a href="#">remove(object)</a>	<a href="#">ICollection</a>	Removes a child object from the collection.  Inherited from <a href="#">ICollection</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IControl</a> .

**Fields details****data**

```
{data.Manager} data
```

Drop-down list data. Names of fields that are available via the `data.Manager.get` method:

- `content` - Title of the drop-down list.
- `title` - Text of the popup hint that appears when the mouse cursor hovers over the list.

**Example:**

```
// Adding a drop-down list to the map and changing its hint
// depending on whether the list is collapsed or expanded.

// Creating a group of event listeners.
var listBoxListener = listBox.events.group()
  .add('expand', function () {
    listBox.data.set('title', 'List is expanded');
  })
  .add('collapse', function () {
    listBox.data.set('title', 'List is collapsed');
  });
map.controls.add(listBox, {float: 'none', top: 10, left: 10});
// ...
map.controls.remove(listBox);
// After deleting the item from the map, we delete the listeners.
```

```
listBoxListener.removeAll();
```

## state

```
{data.Manager} state
```

State of the drop-down list. Names of fields that are available via the `data.Manager.get` method:

- `expanded` - Flag for whether the list is expanded.
- `size` - The size that is currently set for the list.

## Example:

```
// Creating and adding a list that is initially open.  
var listBox = new ymaps.control.ListBox();  
listBox.state.set('expanded', true);  
map.controls.add(listBox);
```

## Events details

### click

Clicking the list title. Instance of the [Event](#) class.

### collapse

The list is closed. Instance of the [Event](#) class.

### expand

The list is open. Instance of the [Event](#) class.

### press

The event indicating that the button has been pressed. Unlike the click event, it is generated only if the state `isEnabled == true`. Instance of the [Event](#) class.

## Methods details

### collapse

```
{control.ListBox} collapse()
```

Collapses the list.

**Returns** self-reference.

### expand

```
{control.ListBox} expand()
```

Expands the list.

**Returns** self-reference.

### getMap

```
{Map} getMap()
```

**Returns** reference to the map.

**isExpanded**

```
{Boolean} isExpanded()
```

**Returns** a flag for whether the control is in the expanded state.

**control.ListBoxItem**

Extends [ICustomizable](#), [ISelectableControl](#).

Drop-down list item.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
control.ListBoxItem([parameters])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">parameters</a>	—	Type: Object String The parameters of the item or string - the HTML content of the item.
<a href="#">parameters.data</a>	—	Type: Object Item data.
<a href="#">parameters.data.content</a>	—	Type: String Item contents.
<a href="#">parameters.options</a>	—	Type: Object Control options.

Parameter	Default value	Description
<a href="#">parameters.options.layout</a>	'islands#listBoxItemLayout'	<p>Type: Function String</p> <p>Constructor of the control layout which implements the <a href="#">ISelectableControlLayout</a> interface or the layout key in the <a href="#">layout.storage</a>. This is the base layout, which selects which of the sub-layouts to display, depending on the value of the "type" option - <a href="#">options.selectableLayout</a> or <a href="#">options.separatorLayout</a>. The layout constructor is passed an object containing the fields:</p> <ul style="list-style-type: none"> <li>control - Reference to the control.</li> <li>options - Options manager for the control.<a href="#">ListBoxItem.options</a>.</li> <li>data - Control data manager <a href="#">control.ListBoxItem.data</a>.</li> <li>state - Control state manager <a href="#">control.ListBoxItem.state</a>.</li> </ul> <p>The layout's outward appearance changes based on the control's data, state and options. The control, in turn, reacts to layout interface events and changes the values of fields for <a href="#">control.ListBoxItem.state</a> depending on the commands received.</p>
<a href="#">parameters.options.selectableLayout</a>	'islands#listBoxItemSelectableLayout'	<p>Type: Function String</p> <p>Constructor of the list item layout which implements the <a href="#">ISelectableControlLayout</a> interface or the layout key in the <a href="#">layout.storage</a>. Applies to items with the option type='item'. Option for standard implementation of the list item layout.</p>
<a href="#">parameters.options.selectOnClick</a>	true	<p>Type: Boolean</p> <p>Flag that allows automatically selecting a list item when clicked.</p> <ul style="list-style-type: none"> <li>If true, the list item is selected after the click and changes the value of the <a href="#">control.ListBoxItem.state</a> field to 'selected'.</li> <li>If false, the list item's appearance and state do not change after a click.</li> </ul>
<a href="#">parameters.options.separatorLayout</a>	'islands#listBoxItemSeparatorLayout'	<p>Type: Function String</p> <p>Constructor of the list item separator layout, which implements the <a href="#">IControlLayout</a> interface or the layout key in <a href="#">layout.storage</a>. Applies to items with the option type='separator'. Option for standard implementation of the list item layout.</p>

Parameter	Default value	Description
<a href="#">parameters.options.type</a>	'selectable'	Type: String  Type of menu item. Depending on the value of this option, the list item layout instantiates one of the sub-layouts - <a href="#">options.selectableLayout</a> or <a href="#">options.separatorLayout</a> . Possible values: <ul style="list-style-type: none"> <li>'selectable' - The list item is selected by a checkmark to the right of the content.</li> <li>'separator' - Separator.</li> </ul>
<a href="#">parameters.options.visible</a>	true	Type: Boolean  Indicates if the control is displayed.
<a href="#">parameters.state</a>	—	Type: Object  Object describing the state of the menu item.
<a href="#">parameters.state.selected</a>	false	Type: Boolean  Indicates whether the item is selected.

## Fields

Name	Type	Description
<a href="#">data</a>	<a href="#">data.Manager</a>	Data for a list item. Names of fields that are available via the <a href="#">data.Manager.get</a> method: <ul style="list-style-type: none"> <li>content - List item content in HTML format.</li> <li>title - Pop-up hint text.</li> </ul>
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">state</a>	<a href="#">data.Manager</a>	State of a list item. Names of fields that are available via the <a href="#">data.Manager.get</a> method: <ul style="list-style-type: none"> <li>selected - Indicates whether the list item is selected.</li> <li>enabled - Indicates whether the list item is active.</li> </ul>

## Events

Name	Description
<a href="#">click</a>	Selecting a list item.
<a href="#">deselect</a>	The control is not selected. Inherited from <a href="#">ISelectableControl</a> .
<a href="#">disable</a>	The control is unavailable. Inherited from <a href="#">ISelectableControl</a> .

Name	Description
<a href="#">enable</a>	The control is available. Inherited from <a href="#">ISelectableControl</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .
<a href="#">select</a>	The control is selected. Inherited from <a href="#">ISelectableControl</a> .

## Methods

Name	Returns	Description
<a href="#">deselect()</a>		Cancels selection of the control (turns it off). Inherited from <a href="#">ISelectableControl</a> .
<a href="#">disable()</a>		Makes the control unavailable (user actions are not allowed). Inherited from <a href="#">ISelectableControl</a> .
<a href="#">enable()</a>		Makes the control available (user actions are allowed). Inherited from <a href="#">ISelectableControl</a> .
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map.
<a href="#">getParent()</a>	<a href="#">IControlParent</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IControl</a> .
<a href="#">isEnabled()</a>	Boolean	Returns true if the control is available, or false if it is unavailable. Inherited from <a href="#">ISelectableControl</a> .
<a href="#">isSelected()</a>	Boolean	Returns true if the control is selected, or false if it is not selected. Inherited from <a href="#">ISelectableControl</a> .

Name	Returns	Description
<code>select()</code>		Selects (turns on) the control. Inherited from <a href="#">ISelectableControl</a> .
<code>setParent(parent)</code>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IControl</a> .

## Fields details

### data

```
{data.Manager} data
```

Data for a list item. Names of fields that are available via the `data.Manager.get` method:

- `content` - List item content in HTML format.
- `title` - Pop-up hint text.

### state

```
{data.Manager} state
```

State of a list item. Names of fields that are available via the `data.Manager.get` method:

- `selected` - Indicates whether the list item is selected.
- `enabled` - Indicates whether the list item is active.

## Events details

### click

Selecting a list item.

## Methods details

### getMap

```
{Map} getMap()
```

**Returns** reference to the map.

## control.Manager

Manager for map controls.

[Constructor](#) | [Fields](#) | [Methods](#)

### Constructor

```
control.Manager(map[, controls[, options]])
```

### Parameters:

Parameter	Default value	Description
<code>map</code> *	—	Type: <a href="#">Map</a> Instance of the map.
<code>controls</code>	—	Type: <a href="#">String[]</a>   <a href="#">IControl[]</a> The controls which should be initially placed on the map.
<code>options</code>	—	Type: <a href="#">Object</a> Manager options.
<code>options.margin</code>	10	Type: <a href="#">Number</a> Distance between adjacent controls and the indent from the edges of the map. Specified in pixels.
<code>options.pane</code>	—	Type: <a href="#">IPane</a> Container for controls.
<code>options.states</code>	['small', 'medium', 'large']	Type: <a href="#">String[]</a> The array of sizes, from smallest to largest.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<code>events</code>	<a href="#">event.Manager</a>	Event manager.
<code>options</code>	<a href="#">option.Manager</a>	Manager options.
<code>state</code>	<a href="#">data.Manager</a>	Manager state. Names of fields that are available via the <code>data.Manager.get</code> method: <ul style="list-style-type: none"> <li><code>size</code> - The state of the controls.</li> </ul>

## Methods

Name	Returns	Description
<code>add(control[, options])</code>	<a href="#">control.Manager</a>	Adds a control to the manager.
<code>each(callback, context)</code>	<a href="#">control.Manager</a>	Calls a handler function for all the controls.
<code>get(index)</code>	<a href="#">IControl</a>  null	Returns the control, or null if no item has been found.
<code>getChildElement(control)</code>	<a href="#">vow.Promise</a>	Returns the Promise object, which is confirmed by the HTML element that should hold the child element.

Name	Returns	Description
<a href="#">getContainer()</a>	HTMLElement	Returns container where elements of the control will be added.
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map.
<a href="#">indexOf(childToFind)</a>	Integer	Returns -1 if the control has not been found, or the index of the item in the manager.
<a href="#">remove(control)</a>	<a href="#">control.Manager</a>	Removing a control from the manager.

## Fields details

### events

```
{event.Manager} events
```

Event manager.

### options

```
{option.Manager} options
```

Manager options.

### state

```
{data.Manager} state
```

Manager state. Names of fields that are available via the `data.Manager.get` method:

- `size` - The state of the controls.

## Methods details

### add

```
{control.Manager} add(control[, options])
```

Adds a control to the manager.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>control *</code>	—	<p>Type: <a href="#">IControl</a> String</p> <p>Controls that are set by instances of classes that implement the <a href="#">IControl</a> interface, or by keys.</p> <p>Acceptable key values:</p> <ul style="list-style-type: none"> <li>"fullscreenControl" - Button for opening the map in full-screen mode, <a href="#">control.FullscreenControl</a>.</li> <li>"geolocationControl" - Button for defining the user location, <a href="#">control.GeolocationControl</a>.</li> <li>"routeEditor" - Button for enabling or disabling the behavior "route editor" <a href="#">control.RouteEditor</a>.</li> <li>"rulerControl" - Button for enabling or disabling the behavior "ruler" <a href="#">control.RulerControl</a>.</li> <li>"searchControl" - Search box for processing the user's search query <a href="#">control.SearchControl</a>.</li> <li>"trafficControl" - Traffic panel <a href="#">control.TrafficControl</a>.</li> <li>"typeSelector" - Panel for selecting the map type <a href="#">control.TypeSelector</a>.</li> <li>"zoomControl" - Zoom slider <a href="#">control.ZoomControl</a>.</li> </ul> <p>Also, you can specify one of the predefined sets of controls using special keys:</p> <ul style="list-style-type: none"> <li>"smallMapDefaultSet" - Basic set of controls, optimized for small maps and mobile phone screens. It includes the following controls: "zoomControl", "searchControl", "typeSelector", "geolocationControl" and "fullscreenControl". All controls in this set are minimized to buttons with icons.</li> <li>"mediumMapDefaultSet" - Basic set of controls, optimized for medium-sized maps and tablet screens. In addition to the basic set of controls (see above), this set includes "rulerControl" and "trafficControl".</li> <li>"largeMapDefaultSet" - Basic set of controls</li> </ul>

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object  Control options.
<a href="#">options.float</a>	"right"	Type: String  The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map.
<a href="#">options.floatIndex</a>	0	Type: Number  The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none".
<a href="#">options.position</a>	—	Type: Object  Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".
<a href="#">options.position.bottom</a>	'auto'	Type: Number String  Position relative to the bottom edge of the map. Only works with float = none.
<a href="#">options.position.left</a>	'auto'	Type: Number String  Position relative to the left edge of the map. Only works with float = none.
<a href="#">options.position.right</a>	'auto'	Type: Number String  Position relative to the right edge of the map. Only works with float = none.
<a href="#">options.position.top</a>	'auto'	Type: Number String  Position relative to the top edge of the map. Only works with float = none.

\* Mandatory parameter/option.

#### Example:

```
map.controls
  .add('zoomControl')
  .add('typeSelector');
```

### each

```
{control.Manager} each(callback, context)
```

Calls a handler function for all the controls.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>callback</code> *	—	Type: Function  Handler function. Receives a collection item as input. If the function returns the value "false," processing stops.
<code>context</code> *	—	Type: Object  Context of the invoked function.

\* Mandatory parameter/option.

### get

```
{IControl|null} get(index)
```

**Returns** the control, or null if no item has been found.

#### Parameters:

Parameter	Default value	Description
<code>index</code> *	—	Type: Number String  The index of the desired element or key.

\* Mandatory parameter/option.

### getChildElement

```
{vow.Promise} getChildElement(control)
```

**Returns** the Promise object, which is confirmed by the HTML element that should hold the child element.

#### Parameters:

Parameter	Default value	Description
<code>control</code> *	—	Type: <a href="#">IControl</a>  Control.

\* Mandatory parameter/option.

### getContainer

```
{HTMLElement} getContainer()
```

**Returns** container where elements of the control will be added.

### getMap

```
{Map} getMap()
```

**Returns** reference to the map.

### indexOf

```
{Integer} indexOf(childToFind)
```

**Returns** -1 if the control has not been found, or the index of the item in the manager.

#### Parameters:

Parameter	Default value	Description
<code>childToFind</code> *	—	Type: <a href="#">String</a>   <a href="#">Control</a> Control or its key.

\* Mandatory parameter/option.

### remove

```
{control.Manager} remove(control)
```

Removing a control from the manager.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>control</code> *	—	Type: <a href="#">IControl</a>   <a href="#">String</a> The deleted control or its key.

\* Mandatory parameter/option.

## control.RouteButton

Extends [IControl](#), [ICustomizable](#).

The Route Button control: button with route panel in the popup. The key of the control in the storage [control.storage](#) — "routeButtonControl".

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
control.RouteButton([parameters])
```

#### Parameters:

Parameter	Default value	Description
<a href="#">parameters</a>	—	Type: Object  Control parameters.
<a href="#">parameters.lazy</a>	true	Type: Boolean  If set to true, all modules needed to work with routes will be loaded lazily when the user opens the panel the first time.
<a href="#">parameters.options</a>	—	Type: Object  Control options. Use the 'routePanel' prefix to configure underlying <a href="#">IRoutePanel</a> options.
<a href="#">parameters.options.adjustMapMargin</a>	false	Type: Boolean  Whether the control registers its size in the map margins manager <a href="#">map.margin.Manager</a> .
<a href="#">parameters.options.autofocus</a>	true	Type: Boolean  Specifies whether route panel must automatically gain focus when popup opens.
<a href="#">parameters.options.collapseOnBlur</a>	true	Type: Boolean  This flag enables to collapse the panel when the control loses focus.
<a href="#">parameters.options.float</a>	"right"	Type: String  The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged in a line, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map. See also the description of the position option.
<a href="#">parameters.options.floatIndex</a>	0	Type: Number  The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none".
<a href="#">parameters.options.popupAnimate</a>	true	Type: Boolean  This flag indicates whether popup expansion and collapse should be animated.

Parameter	Default value	Description
<a href="#">parameters.options.popupFloat</a>	'auto'	Type: Boolean  Specifies which button edge the popup should stick to. Takes the following values: <ul style="list-style-type: none"> <li>'auto' - The side depends on the size of the map and the size of the popup.</li> <li>'right' - The right side of the popup sticks to the right side of the control.</li> <li>'left' - The left side of the popup sticks to the left side of the control.</li> </ul>
<a href="#">parameters.options.popupWidth</a>	'210px'	Type: String  Specifies CSS width of popup. Width is restricted by 176px below and 400px above.
<a href="#">parameters.options.position</a>	—	Type: Object  Object describing the position of a control.
<a href="#">parameters.options.position.bottom</a>	'auto'	Type: Number String  Position relative to the bottom edge of the map.
<a href="#">parameters.options.position.left</a>	10	Type: Number String  Position relative to the left edge of the map.
<a href="#">parameters.options.position.right</a>	'auto'	Type: Number String  Position relative to the right edge of the map.
<a href="#">parameters.options.position.top</a>	108	Type: Number String  Position relative to the top edge of the map.
<a href="#">parameters.options.size</a>	'auto'	Type: String  Defines the appearance of the control. Takes the following values: <ul style="list-style-type: none"> <li>'auto' - The layout is changed automatically depending on the dimensions of the map and the number of added controls.</li> <li>'small' - The layout displays the icon, regardless of the map size.</li> <li>'medium' - The layout displays only text, regardless of the map size.</li> <li>'large' - The layout always displays both the icon and text, regardless of the map size.</li> </ul>

Parameter	Default value	Description
<a href="#">parameters.options.visible</a>	true	Type: Boolean Indicates if the control is displayed.
<a href="#">parameters.state</a>	—	Type: Object Object describing the state of a control.
<a href="#">parameters.state.expanded</a>	false	Type: Boolean Indicates whether the popup with the panel is expanded.

**Examples:****1.**

```
// Example 1.
// Creating a RouteButton control and adding it to the map.
var routeButton = new ymaps.control.RouteButton({
  options: {
    size: "small"
  }
});
myMap.controls.add(routeButton);
```

**2.**

```
// Example 2.
// Add control to the left corner of the map and set default points of arrival and departure.
myMap.controls.add('routeButtonControl', {
  size: "large",
  float: "left",
  floatIndex: 1000,
});
myMap.controls.get('routeButtonControl').routePanel.state.set({
  fromEnabled: false,
  from: "moscow",
  to: "saint petersburg",
  type: "auto"
});
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">IControl</a> .
<a href="#">routePanel</a>	<a href="#">IRoutePanel</a>	

**Events**

Name	Description
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .

Name	Description
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getParent()</a>	<a href="#">IControlParent</a>  null	<p>Returns link to the parent object, or null if the parent element was not set.</p> <p>Inherited from <a href="#">IControl</a>.</p>
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	<p>Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.</p> <p>Inherited from <a href="#">IControl</a>.</p>

## Fields details

### routePanel

```
{IRoutePanel} routePanel
```

## control.RouteEditor

Extends [control.Button](#).

The "Route editor" control. The key of the control in the storage. [control.storage](#) — "routeEditor".

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
control.RouteEditor([parameters])
```

### Parameters:

Parameter	Default value	Description
<a href="#">parameters</a>	—	<p>Type: Object</p> <p>Control parameters.</p>
<a href="#">parameters.data</a>	—	<p>Type: Object</p> <p>Object describing the data of a control.</p>
<a href="#">parameters.data.image</a>	'routes'	<p>Type: String</p> <p>URL of the button icon.</p>

Parameter	Default value	Description
<a href="#">parameters.data.title</a>	—	Type: String  Text of the popup hint that appears when the mouse cursor hovers over the button.
<a href="#">parameters.options</a>	—	Type: Object  Control options.
<a href="#">parameters.options.adjustMapMargin</a>	false	Type: Boolean  Whether the control registers its size in the map margins manager <a href="#">map.margin.Manager</a> .
<a href="#">parameters.options.float</a>	"right"	Type: String  The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map. See also the description of the position option.
<a href="#">parameters.options.floatIndex</a>	100	Type: Number  The priority of the control positioning. Element with the highest possible priority.
<a href="#">parameters.options.layout</a>	—	Type: <a href="#">ISelectableControlLayout</a>  String  Constructor of the control layout which implements the <a href="#">ISelectableControlLayout</a> interface or the layout key in the <a href="#">layout.storage</a> . The layout constructor is passed an object containing the fields: <ul style="list-style-type: none"> <li>control - Reference to the control.</li> <li>options - Options manager for the control.<a href="#">FullscreenControl.options</a>.</li> <li>data - Data manager for the control.<a href="#">FullscreenControl.data</a> control.</li> <li>state - Control state manager <a href="#">control.FullscreenControl.state</a>.</li> </ul> The layout's outward appearance changes based on the control's data, state and options. The control, in turn, reacts to the interface events of the layout

Parameter	Default value	Description
<a href="#">parameters.options.maxWidth</a>	28	Type: Number Number[]  The maximum width of the control in different states. If a number is specified, it is assumed that the control has the same maximum dimensions in all states. If an array is specified, it will be interpreted as the maximum width in different states from the lesser to the greater. The number of states is set in the instance of the class <a href="#">control.Manager</a> , which is usually a field of <a href="#">Map.controls</a> , via the "states" option. By default, the controls have three states ['small', 'medium', 'large']. By default, the control does not change its size and always looks like a button with an icon.
<a href="#">parameters.options.position</a>	—	Type: Object  Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".
<a href="#">parameters.options.position.bottom</a>	'auto'	Type: Number String  Position relative to the bottom edge of the map.
<a href="#">parameters.options.position.left</a>	'auto'	Type: Number String  Position relative to the left edge of the map.
<a href="#">parameters.options.position.right</a>	'auto'	Type: Number String  Position relative to the right edge of the map.
<a href="#">parameters.options.position.top</a>	'auto'	Type: Number String  Position relative to the top edge of the map.
<a href="#">parameters.options.visible</a>	true	Type: Boolean  Indicates if the control is displayed.
<a href="#">parameters.state</a>	—	Type: Object  Object describing the state of a control.

**Example:**

```
// Adding the control to the map.
map.controls.add('routeEditor');
```

**Fields**

Name	Type	Description
<a href="#">data</a>	<a href="#">data.Manager</a>	<p>Button data. Names of fields that are available via the <code>data.Manager.get</code> method:</p> <ul style="list-style-type: none"> <li><code>image</code> - Button icon, if available.</li> <li><code>content</code> - Button content in HTML format.</li> <li><code>title</code> - Text of the popup hint that appears when the mouse cursor hovers over the button.</li> </ul> <p>Inherited from <a href="#">control.Button</a>.</p>
<a href="#">events</a>	<a href="#">IEventManager</a>	<p>Event manager.</p> <p>Inherited from <a href="#">IEventEmitter</a>.</p>
<a href="#">options</a>	<a href="#">IOptionManager</a>	<p>Options manager.</p> <p>Inherited from <a href="#">IControl</a>.</p>
<a href="#">press</a>		<p>The event indicating that the button has been pressed. Unlike the <code>click</code> event, it is generated only if the state <code>isEnabled == true</code>. Instance of the <code>Event</code> class.</p> <p>Inherited from <a href="#">control.Button</a>.</p>
<a href="#">state</a>	<a href="#">data.Manager</a>	<p>Button state. Names of fields that are available via the <code>data.Manager.get</code> method:</p> <ul style="list-style-type: none"> <li><code>selected</code> - Flag for whether the button is selected.</li> <li><code>enabled</code> - Flag for whether the button is active.</li> <li><code>size</code> - The size that is currently set for the button.</li> </ul> <p>Inherited from <a href="#">control.Button</a>.</p>

**Events**

Name	Description
<a href="#">click</a>	<p>Clicking the button. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">control.Button</a>.</p>
<a href="#">deselect</a>	<p>The control is not selected.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">disable</a>	<p>The control is unavailable.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">enable</a>	<p>The control is available.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li><code>oldParent</code> - Old parent.</li> <li><code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>

Name	Description
<a href="#">select</a>	The control is selected. Inherited from <a href="#">ISelectableControl</a> .

## Methods

Name	Returns	Description
<a href="#">deselect()</a>		<p>Cancels selection of the control (turns it off).</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">disable()</a>		<p>Makes the control unavailable (user actions are not allowed).</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">enable()</a>		<p>Makes the control available (user actions are allowed).</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">getMap()</a>	<a href="#">Map</a>	<p>Returns reference to the map.</p> <p>Inherited from <a href="#">control.Button</a>.</p>
<a href="#">getParent()</a>	<a href="#">IControlParent</a>  null	<p>Returns link to the parent object, or null if the parent element was not set.</p> <p>Inherited from <a href="#">IControl</a>.</p>
<a href="#">getRoute()</a>	<a href="#">router.Route</a>	Returns route.
<a href="#">isEnabled()</a>	Boolean	<p>Returns true if the control is available, or false if it is unavailable.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">isSelected()</a>	Boolean	<p>Returns true if the control is selected, or false if it is not selected.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">select()</a>		<p>Selects (turns on) the control.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	<p>Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.</p> <p>Inherited from <a href="#">IControl</a>.</p>

## Methods details

### getRoute

```
{router.Route} getRoute()
```

Returns route.

## control.RoutePanel

Extends [IControl](#), [ICustomizable](#).

The Route Panel control. The key of the control in the storage [control.storage](#) — "routePanelControl".

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
control.RoutePanel([parameters])
```

#### Parameters:

Parameter	Default value	Description
<a href="#">parameters</a>	—	Type: Object  Control parameters.
<a href="#">parameters.options</a>	—	Type: Object  Control options. Use prefix 'routePanel' to configure underlying <a href="#">IRoutePanel</a> options. 'routePanelAdjustMapMargin' option is set to true by default.
<a href="#">parameters.options.autofocus</a>	true	Type: Boolean  Whether panel must automatically gain focus after it was added to map.
<a href="#">parameters.options.float</a>	"left"	Type: String  The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map. See also the description of the position option.
<a href="#">parameters.options.floatIndex</a>	0	Type: Number  The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none".

Parameter	Default value	Description
<a href="#">parameters.options.maxWidth</a>	'210px'	Type: String  Specifies CSS width of popup. Width is restricted by 176px below and 400px above.
<a href="#">parameters.options.position</a>	—	Type: Object  Object describing the position of a control.
<a href="#">parameters.options.position.bottom</a>	—	Type: Number String  Position relative to the bottom edge of the map.
<a href="#">parameters.options.position.left</a>	—	Type: Number String  Position relative to the left edge of the map.
<a href="#">parameters.options.position.right</a>	—	Type: Number String  Position relative to the right edge of the map.
<a href="#">parameters.options.position.top</a>	—	Type: Number String  Position relative to the top edge of the map.
<a href="#">parameters.options.showHeader</a>	false	Type: Boolean  Whether to show header.
<a href="#">parameters.options.title</a>	'Routes'	Type: String  Title to show at the top of the panel. Visible only if showHeader option is true.
<a href="#">parameters.options.visible</a>	true	Type: Boolean  Indicates if the control is displayed.
<a href="#">parameters.state</a>	—	Type: Object  Object describing the state of a control.

**Examples:****1.**

```
// Example 1.
// Adding a route panel control to the map.
myMap.controls.add('routePanelControl')
```

**2.**

```
// Example 2.
// Creating 300px-wide route panel with header with filled starting point.
myMap.controls.add('routePanelControl', {
  maxWidth: 300,
});
var routePanel = myMap.controls.get('routePanelControl').routePanel;
```

```
routePanel.options.set('adjustMapMargin', true);
routePanel.state.set({
  fromEnabled: false,
  from: "moscow",
  to: "saint petersburg",
  type: "auto"
});
```

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">IControl</a> .
<a href="#">routePanel</a>	<a href="#">IRoutePanel</a>	Route panel.

## Events

Name	Description
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .

## Methods

Name	Returns	Description
<a href="#">getParent()</a>	<a href="#">IControlParent</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IControl</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IControl</a> .

## Fields details

### routePanel

```
{IRoutePanel} routePanel
```

Route panel.

## control.RulerControl

Extends [control.Button](#).

The "Ruler" control. The key of the control in the storage. [control.storage](#) — "rulerControl".

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
control.RulerControl([parameters])
```

### Parameters:

Parameter	Default value	Description
<a href="#">parameters</a>	—	Type: Object Control parameters.
<a href="#">parameters.data</a>	—	Type: Object Object describing the data of a control.
<a href="#">parameters.options</a>	—	Type: Object Control options.
<a href="#">parameters.options.adjustMapMargin</a>	false	Type: Boolean Whether the control registers its size in the map margins manager <a href="#">map.margin.Manager</a> .
<a href="#">parameters.options.position</a>	—	Type: Object Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".
<a href="#">parameters.options.position.bottom</a>	30	Type: Number String Position relative to the bottom edge of the map.
<a href="#">parameters.options.position.left</a>	'auto'	Type: Number String Position relative to the left edge of the map.
<a href="#">parameters.options.position.right</a>	10	Type: Number String Position relative to the right edge of the map.
<a href="#">parameters.options.position.top</a>	'auto'	Type: Number String Position relative to the top edge of the map.
<a href="#">parameters.options.scaleLine</a>	true	Type: Boolean Flags if the ruler control should be displayed to the right of the ruler button.

Parameter	Default value	Description
<a href="#">parameters.options.visible</a>	true	Type: Boolean Indicates if the control is displayed.
<a href="#">parameters.state</a>	—	Type: Object Object describing the state of a control.

## Fields

Name	Type	Description
<a href="#">data</a>	<a href="#">data.Manager</a>	Button data. Names of fields that are available via the <a href="#">data.Manager.get</a> method: <ul style="list-style-type: none"> <li><code>image</code> - Button icon, if available.</li> <li><code>content</code> - Button content in HTML format.</li> <li><code>title</code> - Text of the popup hint that appears when the mouse cursor hovers over the button.</li> </ul> Inherited from <a href="#">control.Button</a> .
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">IControl</a> .
<a href="#">press</a>		The event indicating that the button has been pressed. Unlike the <code>click</code> event, it is generated only if the state <code>isEnabled == true</code> . Instance of the <code>Event</code> class. Inherited from <a href="#">control.Button</a> .
<a href="#">state</a>	<a href="#">data.Manager</a>	Button state. Names of fields that are available via the <a href="#">data.Manager.get</a> method: <ul style="list-style-type: none"> <li><code>selected</code> - Flag for whether the button is selected.</li> <li><code>enabled</code> - Flag for whether the button is active.</li> <li><code>size</code> - The size that is currently set for the button.</li> </ul> Inherited from <a href="#">control.Button</a> .

## Events

Name	Description
<a href="#">click</a>	Clicking the button. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">control.Button</a> .
<a href="#">deselect</a>	The control is not selected. Inherited from <a href="#">ISelectableControl</a> .
<a href="#">disable</a>	The control is unavailable. Inherited from <a href="#">ISelectableControl</a> .
<a href="#">enable</a>	The control is available. Inherited from <a href="#">ISelectableControl</a> .

Name	Description
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .
<a href="#">select</a>	The control is selected. Inherited from <a href="#">ISelectableControl</a> .

## Methods

Name	Returns	Description
<a href="#">deselect()</a>		<p>Cancels selection of the control (turns it off).</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">disable()</a>		<p>Makes the control unavailable (user actions are not allowed).</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">enable()</a>		<p>Makes the control available (user actions are allowed).</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">getMap()</a>	<a href="#">Map</a>	<p>Returns reference to the map.</p> <p>Inherited from <a href="#">control.Button</a>.</p>
<a href="#">getParent()</a>	<a href="#">IControlParent</a>  null	<p>Returns link to the parent object, or null if the parent element was not set.</p> <p>Inherited from <a href="#">IControl</a>.</p>
<a href="#">isEnabled()</a>	Boolean	<p>Returns true if the control is available, or false if it is unavailable.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">isSelected()</a>	Boolean	<p>Returns true if the control is selected, or false if it is not selected.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>
<a href="#">select()</a>		<p>Selects (turns on) the control.</p> <p>Inherited from <a href="#">ISelectableControl</a>.</p>

Name	Returns	Description
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	<p>Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.</p> <p>Inherited from <a href="#">IControl</a>.</p>

## control.SearchControl

Extends [IControl](#), [ICustomizable](#).

The "Search on map" control. Allows you to process a user's search query and display the result in the panel and on the map.

Each search result represents a two-line block on the panel of a control. To generate the block, the "name" and "description" fields from the geocoding result object are used.

Key for the control in [control.storage](#) — "searchControl".

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
control.SearchControl([parameters])
```

### Parameters:

Parameter	Default value	Description
<a href="#">parameters</a>	—	<p>Type: Object</p> <p>Control parameters.</p>
<a href="#">parameters.data</a>	—	<p>Type: Object</p> <p>Object describing the data of a control.</p>
<a href="#">parameters.options</a>	—	<p>Type: Object</p> <p>Control options.</p>
<a href="#">parameters.options.adjustMapMargin</a>	false	<p>Type: Boolean</p> <p>Whether the control registers its size in the map margins manager <a href="#">map.margin.Manager</a>.</p>
<a href="#">parameters.options.boundedBy</a>	—	<p>Type: Number[][]</p> <p>A rectangular area on the map, where the object being searched for is presumably located. For ranking, the objects located inside the specified area will receive higher priority.</p>
<a href="#">parameters.options.fitMaxWidth</a>	false	<p>Type: Boolean</p> <p>Whether to stretch control by the maximum width.</p>

Parameter	Default value	Description
<a href="#">parameters.options.float</a>	"right"	<p>Type: String</p> <p>The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map. See also the description of the position option.</p>
<a href="#">parameters.options.floatIndex</a>	200	<p>Type: Number</p> <p>The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none".</p>
<a href="#">parameters.options.formLayout</a>	'islands#searchControlFormLayout'	<p>Type: <a href="#">ILayout</a> String</p> <p>Constructor of the form layout for searching in the default control layout or the key in storage <a href="#">layout.storage</a>.</p> <p>The layout constructor is passed an object containing the fields:</p> <ul style="list-style-type: none"> <li>control - Reference to the control.</li> <li>options - Control options manager <a href="#">control.SearchControl.options</a>.</li> <li>data - Control data manager <a href="#">control.SearchControl.data</a>.</li> <li>state - Control state manager <a href="#">control.SearchControl.state</a>.</li> </ul>
<a href="#">parameters.options.kind</a>	'house'	<p>Type: String</p> <p>Type of toponym (only for reverse geocoding). List of acceptable values:</p> <ul style="list-style-type: none"> <li>house - House or building.</li> <li>street - Street.</li> <li>metro - Subway station.</li> <li>district - City district.</li> <li>locality - City, town, village, etc.</li> </ul>

Parameter	Default value	Description
<a href="#">parameters.options.layout</a>	'islands#searchControlLayout'	<p>Type: <a href="#">ISearchControlLayout</a> String</p> <p>Control layout.</p> <p>The layout constructor is passed an object containing the fields:</p> <ul style="list-style-type: none"> <li>control - Reference to the control.</li> <li>options - Control options manager <a href="#">control.SearchControl.options</a>.</li> <li>data - Control data manager <a href="#">control.SearchControl.data</a>.</li> <li>state - Control state manager <a href="#">control.SearchControl.state</a>.</li> </ul> <p>The layout's outward appearance changes based on the control's data, state and options. The control, in turn, reacts to layout interface events and changes the values of fields for <a href="#">control.SearchControl.state</a> depending on the commands received.</p>
<a href="#">parameters.options.maxWidth</a>	[30, 72, 315]	<p>Type: Number Number[]</p> <p>The maximum width of the control in different states. If a number is specified, it is assumed that the control has the same maximum dimensions in all states. If an array is specified, it will be interpreted as the maximum width in different states from the lesser to the greater. The number of states is set in the instance of the class <a href="#">control.Manager</a>, which is usually a field of <a href="#">Map.controls</a>, via the "states" option. Default search control layout width for large state can be set in range from 280 to 660 pixels. By default, the controls have three states ['small', 'medium', 'large'].</p>
<a href="#">parameters.options.noCentering</a>	false	<p>Type: Boolean</p> <p>If false, the map center is automatically placed so that the object is entirely visible; if true, the map center is not changed when showing the found object. If <code>noCentering = true</code> and <code>noPlacemark = true</code> are set, no visible changes will occur on the map when search results are clicked.</p>

Parameter	Default value	Description
<a href="#">parameters.options.noPlacemark</a>	false	Type: Boolean  If false, a placemark with an open balloon is automatically added to the center of the found object; if true, it is not. If noCentering = true and noPlacemark = true are set, no visible changes will occur on the map when search results are clicked. This option doesn't work with the <code>yandex#search</code> provider.
<a href="#">parameters.options.noPopup</a>	false	Type: Boolean  If true, the drop-down list of results is not shown; if false, it is shown.
<a href="#">parameters.options.noSelect</a>	false	Type: Boolean  If false, the search result will be automatically displayed in case it is the only object found; if true, the result will not be selected.
<a href="#">parameters.options.noSuggestPanel</a>	false	Type: Boolean  If true, the panel with the search suggestions is not shown; if false, it is shown.
<a href="#">parameters.options.placeholderContent</a>	—	Type: String  Text of the hint, displayed in the input field of the control.
<a href="#">parameters.options.popupItemLayout</a>	'islands#searchControlPopupItemLayout'	Type: <a href="#">ILayout</a>  String  Layout constructor for a search result in the drop-down list in the default control layout or the key in storage <a href="#">layout.storage</a> .  The layout constructor is passed an object containing the fields: <ul style="list-style-type: none"> <li>• control - Reference to the control.</li> <li>• options - Control options manager <code>control.SearchControl.options</code>.</li> <li>• data - Control data manager <code>control.SearchControl.data</code>.</li> <li>• state - Control state manager <a href="#">control.SearchControl.state</a>.</li> </ul>

Parameter	Default value	Description
<a href="#">parameters.options.popupLayout</a>	'islands#searchControlPopupLayout'	<p>Type: <a href="#">ILayout</a> String</p> <p>Layout constructor for the drop-down list with search results in the default control layout or the key in storage <a href="#">layout.storage</a>.</p> <p>The layout constructor is passed an object containing the fields:</p> <ul style="list-style-type: none"> <li>control - Reference to the control.</li> <li>options - Control options manager <a href="#">control.SearchControl.options</a>.</li> <li>data - Control data manager <a href="#">control.SearchControl.data</a>.</li> <li>state - Control state manager <a href="#">control.SearchControl.state</a>.</li> </ul>
<a href="#">parameters.options.position</a>	—	<p>Type: Object</p> <p>Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".</p>
<a href="#">parameters.options.position.bottom</a>	'auto'	<p>Type: Number String</p> <p>Position relative to the bottom edge of the map.</p>
<a href="#">parameters.options.position.left</a>	'auto'	<p>Type: Number String</p> <p>Position relative to the left edge of the map.</p>
<a href="#">parameters.options.position.right</a>	'auto'	<p>Type: Number String</p> <p>Position relative to the right edge of the map.</p>
<a href="#">parameters.options.position.top</a>	'auto'	<p>Type: Number String</p> <p>Position relative to the top edge of the map.</p>
<a href="#">parameters.options.provider</a>	'yandex#map'	<p>Type: <a href="#">IGeocodeProvider</a> String</p> <p>Geocoding provider. One of the standard providers can be used:</p> <ul style="list-style-type: none"> <li>'yandex#map' - Search for toponyms on the map.</li> <li>'yandex#search' - Search for toponyms and businesses. When searching for businesses, the following options don't work: "noPlacemark", "noCentering", "noSelect", "strictBounds", "kind". Moreover, if the "results" option is not specified for the control, the map, by default, cannot show more than 20 objects simultaneously.</li> </ul>

Parameter	Default value	Description
<a href="#">parameters.options.searchCoordOrder</a>	'latlong'	Type: String  Determines how to interpret the coordinates in the request. By default, coordinates will be processed as latitude-longitude. This option doesn't work with the <code>yandex#search</code> provider.
<a href="#">parameters.options.size</a>	'auto'	Type: String  Defines the appearance of the control. Takes the following values: <ul style="list-style-type: none"> <li>'small' — Always show the button with the icon.</li> <li>'medium' — Always show the button with text.</li> <li>'large' — Always show a full search form.</li> <li>'auto' — Automatically select the control size, depending on the amount of free space in the toolbar.</li> </ul>
<a href="#">parameters.options.strictBounds</a>	—	Type: Boolean  Search only inside the area defined by the "boundedBy" option. Objects that are outside of the specified area will not be in the output.
<a href="#">parameters.options.suppressYandexSearch</a>	false	Type: Boolean  Whether to hide the message offering to search on the Yandex portal if results were not found.
<a href="#">parameters.options.useMapBounds</a>	—	Type: Boolean  Flag for taking into account the boundaries of the visible map area when searching. If true, the visible area that is calculated has higher priority than the area defined by <code>boundedBy</code> .
<a href="#">parameters.options.zoomMargin</a>	0	Type: Number  Represents the margins from the boundaries of the visible area of the map when displaying search results. The option works only if the value of <code>noCentering</code> is false. If a single number is set, it is applied to each side. If two numbers are set, they are the horizontal and vertical margins, respectively. If an array of four numbers is set, they are the top, right, bottom, and left margins. When the "useMapMargin" option is enabled, the "zoomMargin" value is combined with the values that were calculated in the margins manager <a href="#">map.margin.Manager</a> .

Parameter	Default value	Description
<a href="#">parameters.state</a>	—	Type: Object  Object describing the state of a control.
<a href="#">options.useMapMargin</a>	true	Type: Boolean  Whether to account for map margins <a href="#">map.margin.Manager</a> when showing search results on the map.

**Examples:****1.**

```
// Example 1.
// Creating a control and adding it to the map.

var searchControl = new ymaps.control.SearchControl({
  options: {
    float: 'right',
    floatIndex: 100,
    noPlacemark: true
  }
});
myMap.controls.add(searchControl);
```

**2.**

```
// Example 2.
// If the control was already added to the map using a key from control.storage,
// you can get its instance using the control.Manager.get method
// for control.Manager.

var searchControl = myMap.controls.get('searchControl');
searchControl.events.add('submit', function () {
  console.log('request: ' + searchControl.getRequestString());
}, this);
```

**3.**

```
// Example 3.
// Enabling the business search in a control that is already added to the map.
// If the control was already added to the map using a key from control.storage,
// you can get its instance using the control.Manager.get method.

var searchControl = myMap.controls.get('searchControl');
searchControl.options.set('provider', 'yandex#search');
```

**4.**

```
// Example 4.
// Creating the "map search" control with business search enabled.

var searchControl = new ymaps.control.SearchControl({
  options: {
    float: 'left',
    provider: 'yandex#search'
  }
});
```

**5.**

```
// Example 5.
// Enlarging search control in "large" state.
map.options.set({
  // Default option value is [30, 72, 315],
  // we need to correct only value for "large" state.
  searchControlMaxWidth: [30, 72, 500],
  // Expand large search control to maxWidth.
  fitMaxWidth: true
});
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">IControl</a> .
<a href="#">state</a>	<a href="#">data.Manager</a>	State of the control. Names of fields that are available via the <a href="#">data.Manager.get</a> method: <ul style="list-style-type: none"> <li>size — Current size of the control.</li> <li>results — Array containing search results.</li> <li>currentIndex — Index of the currently selected element.</li> <li>found — Total number of results found.</li> <li>request — Currently active query.</li> <li>correction — Corrected query.</li> <li>noSuggestPanel - Indicates whether the suggestion panel should be hidden.</li> </ul>

**Events**

Name	Description
<a href="#">clear</a>	Event of clearing the search results. Instance of the <a href="#">Event</a> class.
<a href="#">error</a>	Error in getting search results from the server event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>error — Information about the error.</li> </ul>
<a href="#">load</a>	Getting search results from the server event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>skip — How many elements were omitted.</li> <li>count — The number of downloaded items.</li> </ul>
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .
<a href="#">resultselect</a>	"Search result selection" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>index — Index of the selected result.</li> </ul>
<a href="#">resultshow</a>	Event for displaying search results. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>index — Index of the selected result.</li> </ul>
<a href="#">submit</a>	The event of sending a search query to the server. Instance of the <a href="#">Event</a> class.

**Methods**

Name	Returns	Description
<a href="#">clear()</a>		Clears the search results and the contents of the search bar control.
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map.
<a href="#">getParent()</a>	<a href="#">IControlParent</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IControl</a> .
<a href="#">getRequestString()</a>	String	Returns search query.
<a href="#">getResponseMetaData()</a>	Object	Returns the geo search metadata.
<a href="#">getResult(index)</a>	<a href="#">vow.Promise</a>	Getting search results.
<a href="#">getResultsArray()</a>	Object[]	Returns array containing current search results.
<a href="#">getResultsCount()</a>	Integer	Returns total number of results found.
<a href="#">getSelectedIndex()</a>	Integer	Returns index of the selected element.
<a href="#">hideResult()</a>		Hides the result shown on the map.
<a href="#">search(request, options)</a>	<a href="#">vow.Promise</a>	Performs the search.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IControl</a> .
<a href="#">showResult(index)</a>	<a href="#">control.SearchControl</a>	Displays the result with the specified index.

**Fields details****state**

```
{data.Manager} state
```

State of the control. Names of fields that are available via the `data.Manager.get` method:

- `size` — Current size of the control.
- `results` — Array containing search results.
- `currentIndex` — Index of the currently selected element.
- `found` — Total number of results found.
- `request` — Currently active query.
- `correction` — Corrected query.
- `noSuggestPanel` - Indicates whether the suggestion panel should be hidden.

## Events details

### clear

Event of clearing the search results. Instance of the [Event](#) class.

### error

Error in getting search results from the server event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- error — Information about the error.

### load

Getting search results from the server event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- skip — How many elements were omitted.
- count — The number of downloaded items.

### resultselect

"Search result selection" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- index — Index of the selected result.

### resultshow

Event for displaying search results. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- index — Index of the selected result.

### submit

The event of sending a search query to the server. Instance of the [Event](#) class.

## Methods details

### clear

```
{ } clear()
```

Clears the search results and the contents of the search bar control.

### getMap

```
{Map} getMap()
```

**Returns** reference to the map.

### getRequestString

```
{String} getRequestString()
```

**Returns** search query.

### getResponseMetaData

```
{Object} getResponseMetaData()
```

**Returns** the geo search metadata.

### getResult

```
{vow.Promise} getResult(index)
```

Getting search results.

**Returns** object of type [vow.Promise](#).

**Parameters:**

Parameter	Default value	Description
<a href="#">index</a> *	—	Type: Integer  Index of the result, starting from 0.

\* Mandatory parameter/option.

### getResultsArray

```
{Object[]} getResultsArray()
```

**Returns** array containing current search results.

### getResultsCount

```
{Integer} getResultsCount()
```

**Returns** total number of results found.

### getSelectedIndex

```
{Integer} getSelectedIndex()
```

**Returns** index of the selected element.

### hideResult

```
{ } hideResult()
```

Hides the result shown on the map.

**Example:**

```
// If we showed a result on the map using control.SearchControl.showResult,  
// or it was displayed automatically during search, we can hide it, for example,  
// when the button is clicked.  
var myButton = new ymaps.control.Button("Hide results");  
myButton.events.add('click', function () {  
    searchControl.hideResult();  
}, this);  
myMap.controls.add(myButton, { selectOnClick: false });
```

### search

```
{vow.Promise} search(request, options)
```

Performs the search.

**Returns** object of type [vow.Promise](#).

**Parameters:**

Parameter	Default value	Description
<code>request</code> *	—	Type: String Request.
<code>options</code> *	—	Type: Object Additional provider options.

\* Mandatory parameter/option.

#### Example:

```
// Finding Moscow and outputting the "name" field
// from the first result to the console.
searchControl.search('Moscow').then(function () {
  var geoObjectsArray = searchControl.getResultsArray();
  if (geoObjectsArray.length) {
    // Outputs the "name" property of the first geo object in the search results.
    console.log(geoObjectsArray[0].properties.get('name'));
  }
});
```

#### showResult

```
{control.SearchControl} showResult(index)
```

Displays the result with the specified index.

Returns self-reference.

#### Parameters:

Parameter	Default value	Description
<code>index</code> *	—	Type: Integer Index of the result, starting from 0.

\* Mandatory parameter/option.

#### Example:

```
// We want to always show the first result,
// regardless of the number of objects
// found on the map (1 or more).
var searchControl = new ymaps.control.SearchControl({
  // This option disables automatically selecting search results.
  options: { noSelect: true }
});
searchControl.events.add('load', function (event) {
  // Verifying that this event is not completing results loading,
  // but that at least one result was found for the query.
  if (!event.get('skip') && searchControl.getResultsCount()) {
    searchControl.showResult(0);
  }
});
```

## control.storage

Static object.

Instance of [util.Storage](#)

Storage for map controls. Defines control keys with their constructors.

By default, the following controls are added to the storage:

- "rulerControl" - Ruler and scale line [control.RulerControl](#).

- "searchControl" - Search box [control.SearchControl](#).
- "trafficControl" - Traffic panel [control.TrafficControl](#).
- "typeSelector" - Panel for selecting the map type [control.TypeSelector](#).
- "zoomControl" - Zoom slider [control.ZoomControl](#).
- "geolocationControl" - Geolocation control [control.GeolocationControl](#).
- "routeEditor" - Route editor [control.RouteEditor](#).
- "fullscreenControl" - Control for full-screen mode [control.FullscreenControl](#).
- "routeButtonControl" - Button with route panel popup [control.RouteButton](#).
- "routePanelControl" - The Route Panel control [control.RoutePanel](#).

## Methods

### Methods

Name	Returns	Description
<a href="#">add(key, object)</a>	<a href="#">util.Storage</a>	Adds an object to storage.
<a href="#">get(key)</a>	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
<a href="#">remove(key)</a>	<a href="#">util.Storage</a>	Deletes the "key: value" pair from storage.

## control.TrafficControl

Extends [IControl](#), [ICustomizable](#).

The traffic control panel on the map.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
control.TrafficControl([parameters])
```

### Parameters:

Parameter	Default value	Description
<a href="#">parameters</a>	—	Type: Object Control parameters.
<a href="#">parameters.options</a>	—	Type: Object Control options.
<a href="#">parameters.options.adjustMapMargin</a>	false	Type: Boolean Whether the control registers its size in the map margins manager <a href="#">map.margin.Manager</a> .
<a href="#">parameters.options.collapseOnBlur</a>	true	Type: Boolean This flag enables to collapse the panel when the control loses focus. For example, when a user clicks on the document.

Parameter	Default value	Description
<a href="#">parameters.options.float</a>	"right"	<p>Type: String</p> <p>The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map. See also the description of the position option.</p>
<a href="#">parameters.options.floatIndex</a>	100	<p>Type: Number</p> <p>The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none".</p>
<a href="#">parameters.options.layout</a>	—	<p>Type: Function String</p> <p>Control layout. The layout constructor is passed an object containing the fields:</p> <ul style="list-style-type: none"> <li>control - Reference to the control.</li> <li>options - Control options manager for the control. <a href="#">TrafficControl.options</a> control.</li> <li>data - Control data manager for the <a href="#">control.TrafficControl.data</a> control.</li> <li>state - Control state manager for the <a href="#">control.TrafficControl.state</a> control.</li> </ul> <p>The layout's outward appearance changes based on the control's data, state and options. The control, in turn, reacts to layout interface events and changes the values of fields for <a href="#">control.TrafficControl.state</a> depending on the commands received. (Type: constructor for an object with the <a href="#">ITrafficControlLayout</a> interface).</p>
<a href="#">parameters.options.maxWidth</a>	[26, 195, 195]	<p>Type: Number Number[]</p> <p>The maximum width of the button in different states. If a number is specified, it is assumed that the control has the same maximum dimensions in all states. If an array is specified, it will be interpreted as the maximum width in different states from the lesser to the greater. The number of states is set in the instance of the class <a href="#">control.Manager</a>, which is usually a field of <a href="#">Map.controls</a>, via the "states" option. By default, the controls have three states ['small', 'medium', 'large'].</p>

Parameter	Default value	Description
<a href="#">parameters.options.position</a>	—	Type: Object  Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".
<a href="#">parameters.options.position.bottom</a>	'auto'	Type: Number String  Position relative to the bottom edge of the map.
<a href="#">parameters.options.position.left</a>	'auto'	Type: Number String  Position relative to the left edge of the map.
<a href="#">parameters.options.position.right</a>	'auto'	Type: Number String  Position relative to the right edge of the map.
<a href="#">parameters.options.position.top</a>	'auto'	Type: Number String  Position relative to the top edge of the map.
<a href="#">parameters.options.size</a>	'auto'	Type: String  Defines the appearance of the standard traffic control layout. Takes the following values: <ul style="list-style-type: none"> <li>'auto' - The layout is changed automatically depending on the dimensions of the map and the number of added controls.</li> <li>'small' - The button layout displays the traffic light icon, regardless of the map size.</li> <li>'large' - The button layout always displays both the traffic light icon and text, regardless of the map size.</li> </ul>
<a href="#">parameters.options.visible</a>	true	Type: Boolean  Indicates if the control is displayed.
<a href="#">parameters.state</a>	—	Type: Object  State of the control.
<a href="#">parameters.state.providerKey</a>	'traffic#actual'	Type: String  Key for the provider of traffic info shown on the map. <ul style="list-style-type: none"> <li>'traffic#actual' - Traffic "right now".</li> <li>'traffic#archive' - Traffic "normally".</li> </ul>

Parameter	Default value	Description
<a href="#">parameters.state.trafficShown</a>	false	Type: Boolean  Whether traffic data is shown on the map.

**Example:**

```
// Adding the traffic control to the map
// with "current" traffic enabled.
var trafficControl = new ymaps.control.TrafficControl({state: {trafficShown: true}});
map.controls.add(trafficControl, {top: 10, left: 10});
```

**Fields**

Name	Type	Description
<a href="#">data</a>	<a href="#">data.Manager</a>	Panel data.
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">IControl</a> .
<a href="#">state</a>	<a href="#">data.Manager</a>	State of the panel. Names of fields that are available via the <a href="#">data.Manager.get</a> method: <ul style="list-style-type: none"> <li><a href="#">trafficShown</a> - Flag for whether the traffic provider is shown on the map.</li> <li><a href="#">providerKey</a> - Key of the provider that the panel shows. Accepts the values 'traffic#actual' and 'traffic#archive'.</li> <li><a href="#">expanded</a> - Flag for whether the panel is expanded.</li> </ul>

**Events**

Name	Description
<a href="#">collapse</a>	The traffic panel is collapsed. Instance of the <a href="#">Event</a> class.
<a href="#">expand</a>	The traffic panel is expanded. Instance of the <a href="#">Event</a> class.
<a href="#">hidetraffic</a>	Traffic is hidden. Instance of the <a href="#">Event</a> class.
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li><a href="#">oldParent</a> - Old parent.</li> <li><a href="#">newParent</a> - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .
<a href="#">providerkeychange</a>	The provider key changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><a href="#">newProviderKey</a> - New value for the provider key.</li> <li><a href="#">oldProviderKey</a> - Old key value.</li> </ul>
<a href="#">showtraffic</a>	Traffic is shown. Instance of the <a href="#">Event</a> class.

**Methods**

Name	Returns	Description
<a href="#">collapse()</a>		Collapse the traffic panel.
<a href="#">expand()</a>		Expand the traffic panel.
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map.
<a href="#">getParent()</a>	<a href="#">IControlParent</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IControl</a> .
<a href="#">getProvider([key])</a>	<a href="#">ITrafficProvider</a>	Returns instance of the traffic provider.
<a href="#">hideTraffic()</a>		Hide the traffic provider from the map.
<a href="#">isExpanded()</a>	Boolean	Returns flag for whether the panel is expanded.
<a href="#">isTrafficShown()</a>	Boolean	Returns the flag for whether the panel is shown.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IControl</a> .
<a href="#">showTraffic()</a>		Show the traffic provider on the map.

**Fields details****data**

```
{data.Manager} data
```

Panel data.

**state**

```
{data.Manager} state
```

State of the panel. Names of fields that are available via the `data.Manager.get` method:

- `trafficShown` - Flag for whether the traffic provider is shown on the map.
- `providerKey` - Key of the provider that the panel shows. Accepts the values 'traffic#actual' and 'traffic#archive'.
- `expanded` - Flag for whether the panel is expanded.

**Events details****collapse**

The traffic panel is collapsed. Instance of the [Event](#) class.

**expand**

The traffic panel is expanded. Instance of the [Event](#) class.

**hidetraffic**

Traffic is hidden. Instance of the [Event](#) class.

**providerkeychange**

The provider key changed. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `newProviderKey` - New value for the provider key.
- `oldProviderKey` - Old key value.

**showtraffic**

Traffic is shown. Instance of the [Event](#) class.

**Methods details****collapse**

```
{ } collapse()
```

Collapse the traffic panel.

**expand**

```
{ } expand()
```

Expand the traffic panel.

**getMap**

```
{Map} getMap()
```

**Returns** reference to the map.

**getProvider**

```
{ITrafficProvider} getProvider([key])
```

**Returns** instance of the traffic provider.

**Parameters:**

Parameter	Default value	Description
<a href="#">key</a>	—	Type: String  Key of the provider of traffic information. List of available keys: <ul style="list-style-type: none"><li>'traffic#actual' - Provider for traffic "right now".</li><li>'traffic#archive' - Provider for "normal" traffic.</li></ul> If the parameter is omitted, the current provider is returned.

**Example:**

```
// Adding the traffic control to the map.
map.controls.add('trafficControl');
// When opened, the provider for traffic "now" will show the layer of traffic events.
map.controls.get('trafficControl').getProvider('traffic#actual').state.set('infoLayerShown', true);
```

**hideTraffic**

```
{ } hideTraffic()
```

Hide the traffic provider from the map.

**isExpanded**

```
{Boolean} isExpanded()
```

Returns flag for whether the panel is expanded.

**isTrafficShown**

```
{Boolean} isTrafficShown()
```

Returns the flag for whether the panel is shown.

**showTraffic**

```
{ } showTraffic()
```

Show the traffic provider on the map.

**control.TypeSelector**

Extends [control.ListBox](#).

The "Map types" control. For this control, you can add list items that describe map types, as well as additional elements. The key of the control in the storage. [control.storage](#) — "typeSelector".

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
control.TypeSelector([parameters])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">parameters</a>	—	Type: String[] MapType[] Object  Object with descriptions of control parameters. If an array is passed, it is interpreted as an array of map types.
<a href="#">parameters.mapTypes</a>	—	Type: String[] MapType[]  Array of constructors for map types or keys. If the parameter is omitted, the item is added to the standard set of map types. List of available map types: <ul style="list-style-type: none"> <li>• 'yandex#map' - "Roadmap" map type.</li> <li>• 'yandex#satellite' - "Satellite" map type.</li> <li>• 'yandex#hybrid' - "Hybrid" map type.</li> </ul>
<a href="#">parameters.options</a>	—	Type: Object  Control options.
<a href="#">parameters.options.adjustMapMargin</a>	false	Type: Boolean  Whether the control registers its size in the map margins manager <a href="#">map.margin.Manager</a> .
<a href="#">parameters.options.collapseOnBlur</a>	true	Type: Boolean  This flag enables to collapse the list when the button loses focus. For example, when a user clicks on the document.
<a href="#">parameters.options.collapseTimeout</a>	3000	Type: Number  Time delay, after which the open list closes automatically.
<a href="#">parameters.options.expandOnClick</a>	true	Type: Boolean  Flag that allows automatically expanding/collapsing the list when clicked.
<a href="#">parameters.options.float</a>	"right"	Type: String  The side to which you want to align the control. Can take three values: "left", "right" or "none". If set to "left" or "right", the controls are arranged one by one, starting from the left or right edge of the map, respectively. If set to "none", the controls are positioned according to the values of the left, right, bottom and top options, relative to the boundaries of the map. See also the description of the position option.

Parameter	Default value	Description
<a href="#">parameters.options.floatIndex</a>	200	Type: Number  The priority of the control positioning. The element with highest priority is positioned closer to the map boundary that is specified in the float property. Does not work with float = "none".
<a href="#">parameters.options.layout</a>	—	Type: Function String  Constructor of the control layout which implements the <a href="#">ISelectableControlLayout</a> and <a href="#">IGroupControlLayout</a> interfaces or the layout key in the <a href="#">layout.storage</a> . The layout constructor is passed an object containing the fields: <ul style="list-style-type: none"> <li>control - Reference to the control.</li> <li>options - Control options manager <a href="#">control.ListBox.options</a>.</li> <li>data - Control data manager <a href="#">control.ListBox.data</a>.</li> <li>state - Control state manager <a href="#">control.ListBox.state</a>.</li> </ul> The layout's outward appearance changes based on the control's data, state and options. The control, in turn, reacts to layout interface events and changes the values of fields for <a href="#">control.ListBox.state</a> depending on the commands received.
<a href="#">parameters.options.maxWidth</a>	[30, 65, 85]	Type: Number Number[]  The maximum width of the listbox in different states. If a number is specified, it is assumed that the button has the same maximum dimensions in all states. If an array is specified, it will be interpreted as the maximum width of the button in different states, from the lesser to the greater.
<a href="#">parameters.options.panoramasItemMode</a>	'ifMercator'	Type: String  Shows or hides the "Panoramas" element. Possible values: <ul style="list-style-type: none"> <li>'on' - The "Panorama" element is always shown.</li> <li>'ifMercator' - the 'Panorama' element is only shown if the map projection is the Mercator projection.</li> <li>'off' - The "Panorama" element is never shown.</li> </ul> The "Panoramas" element is available only when using the standard layout.

Parameter	Default value	Description
<a href="#">parameters.options.position</a>	—	Type: Object  Object describing the position of a control. If the position option is set, the float option value is automatically treated as "none".
<a href="#">parameters.options.position.bottom</a>	'auto'	Type: Number String  Position relative to the bottom edge of the map.
<a href="#">parameters.options.position.left</a>	'auto'	Type: Number String  Position relative to the left edge of the map.
<a href="#">parameters.options.position.right</a>	'auto'	Type: Number String  Position relative to the right edge of the map.
<a href="#">parameters.options.position.top</a>	'auto'	Type: Number String  Position relative to the top edge of the map.
<a href="#">parameters.options.visible</a>	true	Type: Boolean  Indicates if the control is displayed.
<a href="#">parameters.state</a>	—	Type: Object  State of the control.
<a href="#">parameters.state.expanded</a>	false	Type: Boolean  Flags if the list is expanded.

**Example:**

```
map.controls.add(new ymaps.control.TypeSelector(['yandex#map', 'yandex#hybrid']));
```

**Fields**

Name	Type	Description
<a href="#">data</a>	<a href="#">data.Manager</a>	Control data.
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">IControl</a> .

Name	Type	Description
<a href="#">state</a>	<a href="#">data.Manager</a>	<p>State of the drop-down list. Names of fields that are available via the <code>data.Manager.get</code> method:</p> <ul style="list-style-type: none"> <li><code>expanded</code> - Flag for whether the list is expanded.</li> <li><code>size</code> - The size that is currently set for the list.</li> </ul> <p>Inherited from <a href="#">control.ListBox</a>.</p>

## Events

Name	Description
<a href="#">add</a>	<p>A child object was added.</p> <p>Inherited from <a href="#">ICollection</a>.</p>
<a href="#">click</a>	<p>Clicking the list title. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">control.ListBox</a>.</p>
<a href="#">collapse</a>	<p>The list is closed. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">control.ListBox</a>.</p>
<a href="#">expand</a>	<p>The list is open. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">control.ListBox</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li><code>oldParent</code> - Old parent.</li> <li><code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">press</a>	<p>The event indicating that the button has been pressed. Unlike the <code>click</code> event, it is generated only if the state <code>isEnabled == true</code>. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">control.ListBox</a>.</p>
<a href="#">remove</a>	<p>A child object was deleted.</p> <p>Inherited from <a href="#">ICollection</a>.</p>

## Methods

Name	Returns	Description
<a href="#">add(object)</a>	<a href="#">ICollection</a>	<p>Adds a child object to the collection.</p> <p>Inherited from <a href="#">ICollection</a>.</p>
<a href="#">addMapType(mapType[, positionIndex])</a>	<a href="#">control.TypeSelector</a>	Adds a map type to the list.
<a href="#">collapse()</a>	<a href="#">control.ListBox</a>	<p>Collapses the list.</p> <p>Inherited from <a href="#">control.ListBox</a>.</p>
<a href="#">expand()</a>	<a href="#">control.ListBox</a>	<p>Expands the list.</p> <p>Inherited from <a href="#">control.ListBox</a>.</p>

Name	Returns	Description
<a href="#">getIterator()</a>	<a href="#">Iterator</a>	Returns iterator for the collection. Inherited from <a href="#">ICollection</a> .
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map.
<a href="#">getParent()</a>	<a href="#">IControlParent</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IControl</a> .
<a href="#">isExpanded()</a>	Boolean	Returns a flag for whether the control is in the expanded state. Inherited from <a href="#">control.ListBox</a> .
<a href="#">remove(object)</a>	<a href="#">ICollection</a>	Removes a child object from the collection. Inherited from <a href="#">ICollection</a> .
<a href="#">removeAllMapTypes()</a>	<a href="#">control.TypeSelector</a>	Deletes all map types from the control.
<a href="#">removeMapType(mapType)</a>	<a href="#">control.TypeSelector</a>	Deletes the map type.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IControl</a> .

## Fields details

### data

```
{data.Manager} data
```

Control data.

## Methods details

### addMapType

```
{control.TypeSelector} addMapType(mapType[, positionIndex])
```

Adds a map type to the list.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<a href="#">mapType</a> *	—	Type: <a href="#">String</a>   <a href="#">MapType</a> Map type.

Parameter	Default value	Description
<a href="#">positionIndex</a>	—	Type: Integer  Position in the list (if omitted, the new map type is added to the end of the list). The list of default positionIndex values for standard map types: <ul style="list-style-type: none"> <li>'yandex#map' - 5;</li> <li>'yandex#satellite' - 10;</li> <li>'yandex#hybrid' - 15.</li> </ul>

\* Mandatory parameter/option.

#### Examples:

##### 1.

```
var typeSelector = new ymaps.control.TypeSelector([]);
typeSelector.addMapType('yandex#map', 1);
typeSelector.addMapType('yandex#hybrid', 0);
```

##### 2.

```
// If a standard set of map types is being used,
// and we want to add our own set from mapType.storage // and insert it between "satellite" and "map".
var typeSelector = myMap.controls.get('typeSelector');
typeSelector.addMapType('my#mapType', 6);
```

### getMap

```
{Map} getMap()
```

Returns reference to the map.

### removeAllMapTypes

```
{control.TypeSelector} removeAllMapTypes()
```

Deletes all map types from the control.

Returns self-reference.

### removeMapType

```
{control.TypeSelector} removeMapType(mapType)
```

Deletes the map type.

Returns self-reference.

#### Parameters:

Parameter	Default value	Description
<a href="#">mapType</a> *	—	Type: String  <a href="#">MapType</a>  Map type.

\* Mandatory parameter/option.

## control.ZoomControl

Extends [IControl](#), [ICustomizable](#).

The "Zoom" control. The key of the control in the storage. [control.storage](#) — "zoomControl".

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
control.ZoomControl([parameters])
```

### Parameters:

Parameter	Default value	Description
<a href="#">parameters</a>	—	Type: Object Control parameters.
<a href="#">parameters.data</a>	—	Type: Object Control data.
<a href="#">parameters.options</a>	—	Type: Object Control options.
<a href="#">parameters.options.adjustMapMargin</a>	false	Type: Boolean Whether the control registers its size in the map margins manager <a href="#">map.margin.Manager</a> .
<a href="#">parameters.options.layout</a>	—	Type: <a href="#">IZoomControlLayout</a>  String Constructor of the control layout or the layout key in the <a href="#">layout.storage</a> . The layout constructor is passed an object containing the fields: <ul style="list-style-type: none"> <li>control - Reference to the control.</li> <li>options - Control options manager <a href="#">control.ZoomControl.options</a>.</li> <li>data - Control data manager <a href="#">control.ZoomControl.data</a>.</li> <li>state - Control state manager <a href="#">control.ZoomControl.state</a>.</li> </ul>
<a href="#">parameters.options.position</a>	—	Type: Object Object describing the position of a control.
<a href="#">parameters.options.position.bottom</a>	'auto'	Type: Number String Position relative to the bottom edge of the map.
<a href="#">parameters.options.position.left</a>	10	Type: Number String Position relative to the left edge of the map.

Parameter	Default value	Description
<a href="#">parameters.options.position.right</a>	'auto'	Type: Number String  Position relative to the right edge of the map.
<a href="#">parameters.options.position.top</a>	108	Type: Number String  Position relative to the top edge of the map.
<a href="#">parameters.options.size</a>	'auto'	Type: String  Defines the appearance of the control. Takes the following values: <ul style="list-style-type: none"> <li>'small' — Always show a small zoom control.</li> <li>'large' — Always show a large zoom control.</li> <li>'auto' — Automatically select the size of the control depending on the height of the map container.</li> </ul>
<a href="#">parameters.options.visible</a>	true	Type: Boolean  Indicates if the control is displayed.
<a href="#">parameters.options.zoomDuration</a>	200	Type: Number  The length of animation playback when changing the zoom level.
<a href="#">parameters.options.zoomStep</a>	1	Type: Number  Step of the zoom level change.
<a href="#">parameters.state</a>	—	Type: Object  Object describing the state of a control.

**Examples:****1.**

```
// Example 1.
// Creating a small zoom control and adding it to the map.
var zoomControl = new ymaps.control.ZoomControl({
  options: {
    size: "small"
  }
});
myMap.controls.add(zoomControl);
```

**2.**

```
// Example 2.
// If the control was already added to the map using a key from control.storage.
myMap.controls.add('zoomControl', {
  size: "large"
});
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">IControl</a> .

**Events**

Name	Description
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .

**Methods**

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map.
<a href="#">getParent()</a>	<a href="#">IControlParent</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IControl</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IControl</a> .

**Methods details****getMap**

```
{Map} getMap ()
```

**Returns** reference to the map.

**coordSystem****coordSystem.cartesian**

Static object.

Instance of [ICoordSystem](#)

Class that describes the geometry of a Cartesian plane. Used as the default coordinate system when constructing geodetic lines in non-standard projections.

### Methods

#### Methods

Name	Returns	Description
<a href="#">getDistance(point1, point2)</a>	Number	Returns the shortest distance (along a geodetic line) between the two set points (in meters).
<a href="#">solveDirectProblem(startPoint, direction, distance)</a>	Object	Solves the first (direct) <a href="#">geodesic problem</a> : where we will end up, if we start from a specified point and move in the specified direction for the specified distance, without turning. The following data is a solution for the direct geodetic problem: <ul style="list-style-type: none"> <li>• The end point.</li> <li>• The final direction.</li> <li>• The path function.</li> <li>• A function that allows to specify, for any given moment in time, which point we will be at and which direction we will be moving in.</li> </ul>
<a href="#">solveInverseProblem(startPoint, endPoint[, reverseDirection])</a>	Object	Solves the second (inverse) <a href="#">geodesic problem</a> : construct the shortest route between two points on the mapped surface and determine the distance and direction of movement. Note that on the map of the Earth's surface, the shortest routes are shown as crooked lines. For geo objects in the API, you can enable the mode for displaying shortest distances between points using the "geodesic" option.

## coordSystem.geo

Static object.

Instance of [ICoordSystem](#)

Object that describes the geometry of the Earth's surface. Used for constructing shortest routes (geodetic lines) between points on the Earth's surface and finding distances. Used for displaying geo objects in "geodesic: true" mode.

### Methods

**Methods**

Name	Returns	Description
<a href="#">getDistance(point1, point2)</a>	Number	Returns the shortest distance (along a geodetic line) between the two set points (in meters).
<a href="#">solveDirectProblem(startPoint, direction, distance)</a>	Object	Solves the first (direct) <a href="#">geodesic problem</a> : where we will end up, if we start from a specified point and move in the specified direction for the specified distance, without turning. The following data is a solution for the direct geodetic problem: <ul style="list-style-type: none"> <li>• The end point.</li> <li>• The final direction.</li> <li>• The path function.</li> <li>• A function that allows to specify, for any given moment in time, which point we will be at and which direction we will be moving in.</li> </ul>
<a href="#">solveInverseProblem(startPoint, endPoint[, reverseDirection])</a>	Object	Solves the second (inverse) <a href="#">geodetic problem</a> : construct the shortest route between two points on the mapped surface and determine the distance and direction of movement. Note that on the map of the Earth's surface, the shortest routes are shown as crooked lines. For geo objects in the API, you can enable the mode for displaying shortest distances between points using the "geodesic" option.

**data****data.Manager**

Extends [IDataManager](#), [IFreezable](#).

Custom data manager.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
data.Manager (data)
```

**Parameters:**

Parameter	Default value	Description
<code>data *</code>	—	Type: Object Data.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager for the object. Inherited from <a href="#">IFreezable</a> .

## Events

Name	Description
<code>change</code>	Change to the internal state of the object. Inherited from <a href="#">IFreezable</a> .

## Methods

Name	Returns	Description
<code>freeze()</code>	<a href="#">IFreezable</a>	Switches the object to "frozen" mode. Inherited from <a href="#">IFreezable</a> .
<code>get(path[, defaultValue])</code>	Object	Returns the value of the data field with the specified name.
<code>getAll()</code>	Object	Returns an object containing all the data fields.
<code>isFrozen()</code>	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from <a href="#">IFreezable</a> .
<code>set(path[, value])</code>	<a href="#">data.Manager</a>	Sets values for the specified fields. Two signatures are supported: <ul style="list-style-type: none"> <li>A single argument consisting of a {"name": "value"} object.</li> <li>Two arguments; the first is the field name, and the second is the value.</li> </ul> The name can reference nested fields, i.e. it can contain '..'.
<code>setAll()</code>	<a href="#">data.Manager</a>	Completely overwrites all data fields. Equal to consecutive calls of the "unsetAll" and "set" methods, but with better performance.

Name	Returns	Description
<code>unfreeze()</code>	<code>IFreezable</code>	Switches the object to active mode. Inherited from <code>IFreezable</code> .
<code>unset(path)</code>	<code>data.Manager</code>	Clears the specified data fields.
<code>unsetAll()</code>	<code>data.Manager</code>	Clears all data fields.

## Methods details

### get

```
{Object} get(path[, defaultValue])
```

**Returns** the value of the data field with the specified name.

#### Parameters:

Parameter	Default value	Description
<code>path</code> *	—	Type: String  String with the name of a data field. The name can reference nested fields, i.e. it can contain '!'.  
<code>defaultValue</code>	—	Type: Object  Default value.

\* Mandatory parameter/option.

### getAll

```
{Object} getAll()
```

**Returns** an object containing all the data fields.

### set

```
{data.Manager} set(path[, value])
```

Sets values for the specified fields. Two signatures are supported:

- A single argument consisting of a `{"name": "value"}` object.
- Two arguments; the first is the field name, and the second is the value.

The name can reference nested fields, i.e. it can contain '!'.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>path</code> *	—	Type: Object String  A string containing the name of a data field, or an object of the type {"field name": "value"}.
<code>value</code>	—	Type: Object  The value, if the string containing the field name is passed as the first argument.

\* Mandatory parameter/option.

#### Example:

```
var balloonData = new ymaps.data.Manager({
  balloonContentHeader: 'Balloon title'
});
balloonData.set({
  balloonContentBody: 'Balloon content',
  balloonContentFooter: '&lt;a href=&quot;//ya.ru&quot;&gt;test&lt;/a&gt;'
});
```

#### setAll

```
{data.Manager} setAll()
```

Completely overwrites all data fields. Equal to consecutive calls of the "unsetAll" and "set" methods, but with better performance.

Returns self-reference.

#### Parameters:

Parameter	Default value	Description
<code>data</code> *	—	Type:

\* Mandatory parameter/option.

#### unset

```
{data.Manager} unset(path)
```

Clears the specified data fields.

Returns self-reference.

#### Parameters:

Parameter	Default value	Description
<code>path</code> *	—	Type: String String[]  Field name, or an array of names.

\* Mandatory parameter/option.

**unsetAll**

```
{data.Manager} unsetAll()
```

Clears all data fields.

**Returns** self-reference.

**DomEvent**

Extends [IDomEvent](#).

DOM event in the Yandex.Maps API system. Provides proxy methods for accessing fields and methods of the source DOM event. The returned values are standardized to account for browser variations. The "position" property is also redefined; an array of the type [pageX, pageY] is returned.

[Constructor](#) | [Methods](#)

**Constructor**

```
DomEvent(originalEvent[, type])
```

Creates a DOM event in the Yandex.Maps API system.

**Parameters:**

Parameter	Default value	Description
<a href="#">originalEvent</a> *	—	Type: <a href="#">DomEvent</a> DOM event.
<a href="#">type</a>	—	Type: Object Type of event. If omitted, it is taken from <a href="#">originalEvent.type</a> .

\* Mandatory parameter/option.

**Methods**

Name	Returns	Description
<a href="#">allowMapEvent()</a>		Allows the propagation of the event to the map. Inherited from <a href="#">IEvent</a> .
<a href="#">callMethod(name)</a>		Calls the specified method from the source event. The second and following arguments are passed to the method with the call. Inherited from <a href="#">IEvent</a> .

Name	Returns	Description
<a href="#">get(name)</a>	Object	Returns the value of a property. First it checks whether the property was set via "set", then it checks whether the property exists in "domEvent.overrideStorage". If it's not found, it looks in "originalEvent". Property values are cached.
<a href="#">getSourceEvent()</a>	<a href="#">IDomEvent</a>	Returns source DOM event. Inherited from <a href="#">IDomEvent</a> .
<a href="#">isDefaultPrevented()</a>	Boolean	Returns true if the default reaction to the event has been canceled. Inherited from <a href="#">IEvent</a> .
<a href="#">isImmediatePropagationStopped()</a>	Boolean	Returns true if the event processing has been interrupted. Inherited from <a href="#">IEvent</a> .
<a href="#">isMapEventAllowed()</a>	Boolean	Returns true if the map event is enabled. Inherited from <a href="#">IEvent</a> .
<a href="#">isPropagationStopped()</a>	Boolean	Returns true if event propagation has been interrupted. Inherited from <a href="#">IEvent</a> .
<a href="#">preventDefault()</a>		Cancels the default reaction to an event within the Yandex.Maps API event system. Calling this method does not affect how the browser processes the default action for the source DOM event. Inherited from <a href="#">IDomEvent</a> .
<a href="#">stopImmediatePropagation()</a>		Stops event processing in the Yandex.Maps API event system. I.e. after calling this method, no handler for this event will be called. Calling this method does not affect the processing of the original DOM-event at the browser level. Inherited from <a href="#">IDomEvent</a> .

Name	Returns	Description
<a href="#">stopPropagation()</a>		Stops propagation of the DOM event in the Yandex.Maps API event system. Calling this method does not affect propagation of the source DOM event through the DOM tree.  Inherited from <a href="#">IDomEvent</a> .

## Methods details

### get

```
{Object} get(name)
```

**Returns** the value of a property. First it checks whether the property was set via "set", then it checks whether the property exists in "domEvent.overrideStorage". If it's not found, it looks in "originalEvent". Property values are cached.

#### Parameters:

Parameter	Default value	Description
<a href="#">name</a> *	—	Type: String  Property name.

\* Mandatory parameter/option.

## domEvent

### domEvent.manager

Static object.

Provides a singular interface for working with DOM element events in all browsers and on all devices. For devices that do not have mouse support, events will be mapped to mouse events.

- The touch start event (touchstart/pointerdown) with a single touch point is interpreted as a sequence of "mouseenter", "mousemove" and "mousedown" events.
- The moving touch event (touchmove/pointermove) with a single touch point is mapped to the "mousemove" event.
- Events for ending touch (touchend/pointerup) or canceling touch (touchcancel/pointercancel) are mapped to a sequence of "mouseup", "mousemove", and "mouseleave" events, if a touch start event with a single touch point already occurred earlier.
- A quick succession of start and end events with a single touch point without movement is mapped to a "click" event.
- A quick succession of two "click" events is mapped to a "dblclick" event.
- If there was a lengthy pause between the start and end events with a single touch point and without movement, this is mapped to the "contextmenu" event.

Special events for handling multiple simultaneous touches are also supported.

- "multitouchstart" is sent when touch start events are received with two or more touch points.
- "multitouchmove" is sent when moving touch events are received with two or more touch points.
- "multitouchend" is sent when touch end events are received, if the "multitouchstart" event was sent earlier.

- When adding/removing a touch point, the "multitouchend" event and the "multitouchstart" event will be sent if the remaining number of points is greater than or equal to two.

Manager for working with DOM element events.

## Methods

### Examples:

#### 1.

```
// Listening for events of a single DOM element.
var block = document.getElementById('block');
ymaps.domEvent.manager
  .add(block, 'click', function (event) {
    console.log(event.get('type')); // click
  })
  .add(block, 'mouseleave', function (event) {
    console.log(event.get('type')); // mouseleave
  })
// Setting a single listener for multiple events.
.add(block, ['mousedown', 'mouseup'], function (event) {
  console.log(event.get('type')); // mousedown / mouseup
});
```

#### 2.

```
// Using the events container.
var block = document.getElementById('block');
var domEventsGroup = ymaps.domEvent.manager.group(block);
domEventsGroup.add('click', function (event) {
  console.log(event.get('type')); // click
  // Deleting all event listeners.
  domEventsGroup.removeAll();
});
```

#### 3.

```
// Executing the listener in the context of a specific object.
var block = document.getElementById('block');
// Defining the class.
var someClass = function () {
  this.property = 'value';
};
// Creating a class implementation.
var someObj = new someClass();
ymaps.domEvent.manager.add(block, 'click', function (event) {
  console.log(this.property + ' ' + event.get('type')); // value click
}, someObj);
```

#### 4.

```
// On devices with touch support, we can listen to special multitouch* events
var block = document.getElementById('block');
ymaps.domEvent.manager
  .add(block, ['multitouchstart', 'multitouchmove', 'multitouchend'], function (event) {
    console.log(event.get('type')); // multitouchstart / multitouchmove / multitouchend
    // Not allowing users to move and scale the page using touch.
    event.callMethod('preventDefault');
  });
```

## Methods

Name	Static	Returns	Description
<code>domEvent.manager.add(<a href="#">htmlElement</a>, <a href="#">types</a>, <a href="#">callback</a>, <a href="#">context</a>, <a href="#">capture</a>)</code>		<a href="#">domEvent.manager</a>	Adds a listener for the object's DOM events.
<code>domEvent.manager.group(<a href="#">htmlElement</a>, <a href="#">capture</a>)</code>		<a href="#">event.Group</a>	Returns group of event listeners for the specified DOM element.
<code>domEvent.manager.remove(<a href="#">htmlElement</a>, <a href="#">types</a>, <a href="#">callback</a>, <a href="#">context</a>, <a href="#">capture</a>)</code>		<a href="#">domEvent.manager</a>	Deletes the listener for the objects's DOM events.

## Methods details

### add

```
{domEvent.manager} <static> domEvent.manager.add(htmlElement, types, callback[, context[, capture]])
```

Adds a listener for the object's DOM events.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>htmlElement</code> *	—	Type: HTMLElement Document  The DOM element whose events need to be listened for.
<code>types</code> *	—	Type: String String[]  Event type or types.
<code>callback</code> *	—	Type: Function  Handler function for the event.
<code>context</code>	—	Type: Object  Context for the handler function.
<code>capture</code>	—	Type: Boolean  Indicates if the event should be monitored at the capture phase.

\* Mandatory parameter/option.

### group

```
{event.Group} <static> domEvent.manager.group(htmlElement[, capture])
```

**Returns** group of event listeners for the specified DOM element.

#### Parameters:

Parameter	Default value	Description
<code>htmlElement</code> *	—	Type: HTMLElement Document  DOM element.
<code>capture</code>	—	Type: Boolean  Indicates if the event should be monitored at the capture phase.

\* Mandatory parameter/option.

## remove

```
{domEvent.manager} <static> domEvent.manager.remove(htmlElement, types, callback[, context[, capture]])
```

Deletes the listener for the objects's DOM events.

**Returns** self-reference.

### Parameters:

Parameter	Default value	Description
<code>htmlElement</code> *	—	Type: HTMLElement Document  The DOM element whose events are listened for.
<code>types</code> *	—	Type: String[String[]]  Event type or types.
<code>callback</code> *	—	Type: Function[String]  Handler function for the event, or the unique id of the callback-context pair.
<code>context</code>	—	Type: Object  Context for the handler function.
<code>capture</code>	—	Type: Boolean  Indicates if the event should be monitored at the capture phase.

\* Mandatory parameter/option.

## domEvent.MultiPointer

Extends IMultiTouchEvent.

Object describing the multitouch event that was called by multiple PointerEvent events. Provides proxy methods for accessing fields and methods of DOM events.

[Constructor](#) | [Methods](#)

### Constructor

```
domEvent.MultiPointer(originalEvent[, type])
```

Creates an object that describes a multitouch event.

### Parameters:

Parameter	Default value	Description
<code>originalEvent</code> *	—	Type: Object  pointer event.

Parameter	Default value	Description
<code>type</code>	—	Type: String  Type of event. If omitted, it is assumed to be 'multi' + <code>originalEvent.type</code> .

\* Mandatory parameter/option.

## Methods

Name	Returns	Description
<code>get(name)</code>	Object	Returns the value of the original event property. Property values are cached.

## Methods details

### get

```
{Object} get (name)
```

**Returns** the value of the original event property. Property values are cached.

#### Parameters:

Parameter	Default value	Description
<code>name</code> *	—	Type: String  Property name.

\* Mandatory parameter/option.

## domEvent.MultiTouch

Extends `IMultiTouchEvent`.

Event object. Provides proxy methods for accessing fields and methods of DOM events. The returned values are standardized to account for browser variations.

[Constructor](#) | [Methods](#)

### Constructor

```
domEvent.MultiTouch (originalEvent[, type])
```

Creates an event object that describes a multitouch event.

#### Parameters:

Parameter	Default value	Description
<code>originalEvent</code> *	—	Type: Object  Multitouch event.
<code>type</code>	—	Type: String  Type of event. If omitted, it is assumed to be 'multi' + <code>originalEvent.type</code> .

\* Mandatory parameter/option.

## Methods

Name	Returns	Description
<a href="#">get(name)</a>	Object	Returns the value of the original event property. Property values are cached.

## Methods details

### get

```
{Object} get(name)
```

**Returns** the value of the original event property. Property values are cached.

#### Parameters:

Parameter	Default value	Description
<a href="#">name</a> *	—	Type: String Property name.

\* Mandatory parameter/option.

## domEvent.Pointer

Extends `IMultiTouchEvent`.

Event object. Provides proxy methods for accessing the fields and methods of a DOM event (single touch on the screen).

[Constructor](#) | [Methods](#)

### Constructor

```
domEvent.Pointer(originalEvent[, type])
```

Creates an object that describes a pointer event.

#### Parameters:

Parameter	Default value	Description
<a href="#">originalEvent</a> *	—	Type: Object DOM event.
<a href="#">type</a>	—	Type: String Type of event. If omitted, it is taken from <code>originalEvent.type</code> .

\* Mandatory parameter/option.

**Methods**

Name	Returns	Description
<a href="#">get(name)</a>	Object	Returns the value of the original event property. Property values are cached.

**Methods details****get**

```
{Object} get(name)
```

**Returns** the value of the original event property. Property values are cached.

**Parameters:**

Parameter	Default value	Description
<a href="#">name</a> *	—	Type: String Property name.

\* Mandatory parameter/option.

**domEvent.Touch**

Extends `IMultiTouchEvent`.

Event object. Provides proxy methods for accessing fields and methods of DOM events. The returned values are standardized to account for browser variations.

[Constructor](#) | [Methods](#)

**Constructor**

```
domEvent.Touch(originalEvent[, type])
```

Creates an event object that describes a touch event (single touch on the screen).

**Parameters:**

Parameter	Default value	Description
<a href="#">originalEvent</a> *	—	Type: Object DOM event.
<a href="#">type</a>	—	Type: String Type of event. If omitted, it is taken from <code>originalEvent.type</code> .

\* Mandatory parameter/option.

**Methods**

Name	Returns	Description
<a href="#">get(name)</a>	Object	Returns the value of the original event property. Property values are cached.

## Methods details

### get

```
{Object} get(name)
```

Returns the value of the original event property. Property values are cached.

#### Parameters:

Parameter	Default value	Description
<a href="#">name</a> *	—	Type: String Property name.

\* Mandatory parameter/option.

## event

### event.Group

Extends [IEventGroup](#).

A group of event listeners.

[Constructor](#) | [Fields](#) | [Methods](#)

#### Constructor

```
event.Group(events)
```

Creates a group of event listeners.

#### Parameters:

Parameter	Default value	Description
<a href="#">events</a> *	—	Type: <a href="#">IEventManager</a> The event manager that the group was created for.

\* Mandatory parameter/option.

#### Example:

```
// Creating a group of event listeners.
var listeners = events.group()
  .add('click', function () {
    alert('click');
  })
  .add('dblclick', function () {
    alert('dblclick');
  });
// ...
// When the event handlers stored in the container
// are no longer needed, we simply clear the group.
listeners.removeAll();
```

#### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	The event manager that the group was created for.

**Methods**

Name	Returns	Description
<code>add(types, callback[, context[, priority]])</code>	<code>IEventGroup</code>	Adds an event listener. Inherited from <code>IEventGroup</code> .
<code>getLength()</code>	Integer	Returns the current number of subscribers in the group.
<code>remove(types, callback[, context[, priority]])</code>	<code>IEventGroup</code>	Deletes an event listener from the group. Inherited from <code>IEventGroup</code> .
<code>removeAll()</code>	<code>IEventGroup</code>	Deletes all event listeners from the group. Inherited from <code>IEventGroup</code> .

**Fields details****events**

```
{IEventManager} events
```

The event manager that the group was created for.

**Methods details****getLength**

```
{Integer} getLength()
```

**Returns** the current number of subscribers in the group.

**event.Manager**

Extends `IEventManager`.

Event manager. Using an event manager, you can subscribe to and unsubscribe from events, as well as initiate the events themselves. The manager implements the possibility of building a hierarchy of event propagation using the method `event.Manager.setParent`.

Event propagation has three phases:

- 1. Direct subscribers get events.
- 2. Objects that are higher up in the hierarchy get events when they are relayed on the parent event manager.
- 3. Default action handlers get events via a type + 'defaultaction' service event; the default action is performed only if the event's "target" field matches the context for the event manager.

The manager also allows you to specify the priority when adding event handlers. When throwing events, the handlers will be called in order of decreasing priority.

Subscriptions with the same callback and context parameters but different priority settings are considered to be different. When removing subscriptions, you must specify the same priority as was set when it was added.

[Constructor](#) | [Methods](#)

**Constructor**

```
event.Manager([params])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">params</a>	—	Type: Object Parameters of the event manager.
<a href="#">params.context</a>	—	Type: Object The context object of the event manager.
<a href="#">params.controllers</a>	—	Type: <a href="#">IEventWorkflowController</a> [] Controller or controllers for the event manager.
<a href="#">params.parent</a>	—	Type: <a href="#">IEventManager</a> Parent event manager.

**Methods**

Name	Returns	Description
<a href="#">add(types, callback[, context[, priority]])</a>	<a href="#">IEventManager</a>	Adds a new subscription. Inherited from <a href="#">IEventManager</a> .
<a href="#">createEventObject(type, event, target)</a>	<a href="#">Event</a>	Function that creates the event object. Called in the "fire" method when the passed object is not an instance of the Event class or its descendant.
<a href="#">fire(type[, event])</a>	<a href="#">event.Manager</a>	Fires an event.
<a href="#">getParent()</a>	<a href="#">IEventManager</a>  null	Returns reference to the parent event manager. Inherited from <a href="#">IEventManager</a> .
<a href="#">group()</a>	<a href="#">IEventGroup</a>	Returns the group of event listeners associated with the given event manager. Inherited from <a href="#">IEventManager</a> .
<a href="#">once(types, callback[, context[, priority]])</a>	<a href="#">IEventManager</a>	Adds a listener, which calls the handler function only one time. Inherited from <a href="#">IEventManager</a> .
<a href="#">remove(types, callback[, context[, priority]])</a>	<a href="#">IEventManager</a>	Removes an existing subscription. Inherited from <a href="#">IEventManager</a> .

Name	Returns	Description
<code>setParent(parent)</code>	<code>event.Manager</code>	Sets the parent event manager.

## Methods details

### createEventObject

```
{Event} createEventObject(type, event, target)
```

Function that creates the event object. Called in the "fire" method when the passed object is not an instance of the Event class or its descendant.

**Returns** Event object.

#### Parameters:

Parameter	Default value	Description
<code>type *</code>	—	Type: String Type of event.
<code>event *</code>	—	Type: Object Object that describes the event.
<code>target *</code>	—	Type: Object Object that the event occurred on.

\* Mandatory parameter/option.

### fire

```
{event.Manager} fire(type[, event])
```

Fires an event.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>type *</code>	—	Type: String Type of event.
<code>event</code>	—	Type: Object  <a href="#">Event</a> Event. If a hash with data is passed, the createEventObject method will be called for it and further actions will be performed with the newly created event.

\* Mandatory parameter/option.

**setParent**

```
{event.Manager} setParent (parent)
```

Sets the parent event manager.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>parent *</code>	—	Type: <a href="#">IEventManager</a>  null Parent event manager.

\* Mandatory parameter/option.

**event.Mapper**

Extends [IEventTrigger](#).

Event mapper. Allows managing propagation of events along the hierarchy of event managers.

[Constructor](#) | [Methods](#)

**Constructor**

```
event.Mapper (targetEventManager, mappingTable)
```

**Parameters:**

Parameter	Default value	Description
<code>targetEventManager *</code>	—	Type: <a href="#">IEventManager</a>  The event manger that the mapper propagates initiated events to.
<code>mappingTable *</code>	—	Type: Object  Table of mapping rules. A hash where the keys are types of events and the values are the corresponding mapping functions, or Boolean values. A mapping function for a specific type of event receives an event instance that was initiated on the mapper, and must return the event instance for propagating further along the hierarchy, or null if propagation must be prohibited. The boolean values are interpreted as follows: <ul style="list-style-type: none"> <li>• true - Events of this type are propagated through the hierarchy unchanged.</li> <li>• false - Events of this type are not propagated through the hierarchy.</li> </ul> The "" key is also available in the table for the default processing rule.

\* Mandatory parameter/option.

**Example:**

```
// Creating and setting up the event mapper that will transform the "click" event in the root
// geo object collection into a "geobjectclick" event on the map itself.
var mapper = new ymaps.event.Mapper(myMap.events, {
  "**": false,
  "click": function (event) {
    return new ymaps.Event({
      type: "geobjectclick",
      target: map,
      originalTarget: event.get("target")
    }, event);
  }
});

myMap.geoObjects.events.setParent(mapper);
```

**Methods**

Name	Returns	Description
<a href="#">fire</a> (type[, <a href="#">eventObject</a> ])	<a href="#">IEventTrigger</a>	Triggers an event. Inherited from <a href="#">IEventTrigger</a> .

**Event**

Extends [IEvent](#).

Event. Provides methods for accessing the originalObject object's fields and methods, with the possibility for redefining them.

[Constructor](#) | [Methods](#)

**Constructor**

```
Event(originalEvent [, sourceEvent])
```

Creates an event.

**Parameters:**

Parameter	Default value	Description
<a href="#">originalEvent</a> *	—	Type: Object Source data.
<a href="#">sourceEvent</a>	—	Type: <a href="#">IEvent</a> Source event.

\* Mandatory parameter/option.

**Methods**

Name	Returns	Description
<a href="#">allowMapEvent</a> ()		Allows the propagation of the event to the map. Inherited from <a href="#">IEvent</a> .

Name	Returns	Description
<a href="#">callMethod(name)</a>	Object	<p>Calls the specified method. The operation is equivalent to searching fields via "get" and making a call that passes originalEvent as context. All arguments after the first one are passed as parameters to the method being called.</p>
<a href="#">get(name)</a>	Object	<p>Returns the field value from originalEvent. originalEvent always has the following fields:</p> <ul style="list-style-type: none"><li>• type - String event type.</li><li>• target - Reference to the object that generated the event.</li></ul>
<a href="#">getSourceEvent()</a>	<a href="#">IEvent</a>  null	<p>Returns source event.</p> <p>Inherited from <a href="#">IEvent</a>.</p>
<a href="#">isDefaultPrevented()</a>	Boolean	<p>Checks whether the default reaction to the event is canceled in the Yandex.Maps API event system.</p>
<a href="#">isImmediatePropagationStopped()</a>	Boolean	<p>Checks whether event propagation is stopped in the Yandex.Maps API event system.</p>
<a href="#">isMapEventAllowed()</a>	Boolean	<p>Returns true if the map event is enabled.</p> <p>Inherited from <a href="#">IEvent</a>.</p>
<a href="#">isPropagationStopped()</a>	Boolean	<p>Checks whether event propagation up the hierarchy of objects and collections is stopped in the Yandex.Maps API event system.</p>
<a href="#">preventDefault()</a>		<p>Cancels the default reaction to an event within the Yandex.Maps API event system. Calling this method does not affect propagation of the DOM source event (if there is one) through the DOM tree.</p>
<a href="#">stopImmediatePropagation()</a>		<p>Stops event propagation in the Yandex.Maps API event system. Calling this method does not affect propagation of the DOM source event (if there is one) through the DOM tree.</p>

Name	Returns	Description
<a href="#">stopPropagation()</a>		Stops event propagation up the hierarchy of objects and collections in the Yandex.Maps API event system. Calling this method does not affect propagation of the DOM source event (if there is one) through the DOM tree.

## Methods details

### callMethod

```
{Object} callMethod(name)
```

Calls the specified method. The operation is equivalent to searching fields via "get" and making a call that passes originalEvent as context. All arguments after the first one are passed as parameters to the method being called.

**Returns** value.

**Parameters:**

Parameter	Default value	Description
<a href="#">name</a> *	—	Type: String Method name.

\* Mandatory parameter/option.

### get

```
{Object} get(name)
```

**Returns** the field value from originalEvent. originalEvent always has the following fields:

- type - String event type.
- target - Reference to the object that generated the event.

**Parameters:**

Parameter	Default value	Description
<a href="#">name</a> *	—	Type: String Property name.

\* Mandatory parameter/option.

### Example:

```
// Synchronizing two objects with each other.
object1.events.add(["add", "remove"], function (event) {
  object2[event.get("type")](event.get("child"));
});
```

### isDefaultPrevented

```
{Boolean} isDefaultPrevented()
```

Checks whether the default reaction to the event is canceled in the Yandex.Maps API event system.

**Returns** true if the default reaction to the event is canceled, otherwise false.

### **isImmediatePropagationStopped**

```
{Boolean} isImmediatePropagationStopped()
```

Checks whether event propagation is stopped in the Yandex.Maps API event system.

**Returns** true if propagation was stopped, or false if not.

### **isPropagationStopped**

```
{Boolean} isPropagationStopped()
```

Checks whether event propagation up the hierarchy of objects and collections is stopped in the Yandex.Maps API event system.

**Returns** true if propagation up the hierarchy is canceled; otherwise, false.

### **preventDefault**

```
{ } preventDefault()
```

Cancels the default reaction to an event within the Yandex.Maps API event system. Calling this method does not affect propagation of the DOM source event (if there is one) through the DOM tree.

### **stopImmediatePropagation**

```
{ } stopImmediatePropagation()
```

Stops event propagation in the Yandex.Maps API event system. Calling this method does not affect propagation of the DOM source event (if there is one) through the DOM tree.

### **stopPropagation**

```
{ } stopPropagation()
```

Stops event propagation up the hierarchy of objects and collections in the Yandex.Maps API event system. Calling this method does not affect propagation of the DOM source event (if there is one) through the DOM tree.

## **formatter**

Static object.

Static class containing methods for formatting measurement units depending on the current language.

### [Methods](#)

#### **Methods**

<b>Name</b>	<b>Returns</b>	<b>Description</b>
<a href="#">distance</a> (value[, significantDigits])	String	Returns string representation of distance formatted for the locale being used, and converted to the appropriate measurement system (for example, for en-US, distance will be in miles).

Name	Returns	Description
<code>duration(value[, significantDigits])</code>	String	Returns string representation of duration.

## Methods details

### distance

```
{String} distance(value[, significantDigits])
```

**Returns** string representation of distance formatted for the locale being used, and converted to the appropriate measurement system (for example, for en-US, distance will be in miles).

#### Parameters:

Parameter	Default value	Description
<code>value</code> *	—	Type: Number Length in meters.
<code>significantDigits</code>	2	Type: Integer Number of significant digits in the response.

\* Mandatory parameter/option.

### duration

```
{String} duration(value[, significantDigits])
```

**Returns** string representation of duration.

#### Parameters:

Parameter	Default value	Description
<code>value</code> *	—	Type: Number Duration in seconds.
<code>significantDigits</code>	2	Type: Integer Number of significant digits in the response.

\* Mandatory parameter/option.

## geocode

Static function.

Processes geocoding requests. The request result can be provided in JSON format or as a [GeoObjectCollection](#) object. The geocoder's response format is described in [Geocoding](#).

See [GeocodeResult](#)

**Returns** Promise object.

```
{ vow.Promise } geocode(request[, options])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">request</a> *	—	Type: String Number[]  The address for which coordinates need to be obtained (forward geocoding), or the coordinates for which the address needs to be determined (reverse geocoding).
<a href="#">options</a>	—	Type: Object  Options.
<a href="#">options.boundedBy</a>	—	Type: Number[][]  A rectangular area on the map, where the object being searched for is presumably located.
<a href="#">options.json</a>	false	Type: Boolean  If true, JSON is passed to the handler function. Otherwise, the handler function is passed an object containing the <code>geoObjects</code> field with the geocoding results as <a href="#">GeoObjectCollection</a> . When geocoding using the 'yandex#map' geocoder, the collection contains <a href="#">GeocodeResult</a> objects.
<a href="#">options.kind</a>	'house'	Type: String  Type of toponym (only for reverse geocoding).  List of acceptable values: <ul style="list-style-type: none"> <li>• house - House or building.</li> <li>• street - Street.</li> <li>• metro - Subway station.</li> <li>• district - City district.</li> <li>• locality - City, town, village, etc.</li> </ul>
<a href="#">options.provider</a>	'yandex#map'	Type: <a href="#">IGeocodeProvider</a>  String  Geocoding provider. One of the standard providers can be used: <ul style="list-style-type: none"> <li>• 'yandex#map' - Search on the map.</li> </ul>
<a href="#">options.results</a>	10	Type: Integer  Maximum number of results to be returned.
<a href="#">options.searchCoordOrder</a>	—	Type: String  Determines how to interpret the coordinates in the request.

Parameter	Default value	Description
<code>options.skip</code>	0	Type: Integer Number of results that must be skipped.
<code>options.strictBounds</code>	false	Type: Boolean Search only inside the area defined by the "boundedBy" option.

\* Mandatory parameter/option.

### Examples:

#### 1.

```
// Performs a search for an object named "Moscow".
// The result is immediately displayed on the map.
var myGeocoder = ymaps.geocode("Moscow");
myGeocoder.then(
  function (res) {
    map.geoObjects.add(res.geoObjects);
    // Taking the data resulting from geocoding the object
    // and outputting it to the console.
    console.log(res.geoObjects.get(0).properties.get('metaDataProperty').getAll());
  },
  function (err) {
    // error handling
  }
);
```

#### 2.

```
// Implements the IGeocodeProvider interface.
var randomPointProvider = {
  geocode: function (request, options) {
    var deferred = ymaps.vow.defer(),
        geoObjects = new ymaps.GeoObjectCollection(),
        results = options.results || 10;

    for (var i = 0; i < results; i++) {
      geoObjects.add(new ymaps.GeoObject({
        geometry: {
          type: "Point",
          coordinates: [(Math.random() - 0.5) * 180, (Math.random() - 0.5) * 360]
        },
        properties: {
          name: request + ' ' + i,
          description: request + '\s description ' + i,
          balloonContentBody: '<p>' + request + ' ' + i + '</p>'
        }
      }));
    }

    var response = {
      geoObjects: geoObjects, // search output geo objects
      // Response metainformation.
      metaData: {
        geocoder: {
          request: request, // processed request string
          found: results, // number of results found
          results: results, // number of results returned
          skip: options.skip || 0 // number of results skipped
        }
      }
    };

    // Asynchronous processing.
    setTimeout(function () {
      deferred.resolve(response);
    }, 0);

    return deferred.promise();
  }
};

var myGeocoder = ymaps.geocode("Moscow", { provider: randomPointProvider });

myGeocoder.then(
  function (res) {
    map.geoObjects.add(res.geoObjects);
  },
  function (err) {
    // handling errors
  }
);
```

);

## GeocodeResult

Extends [IGeoObject](#).

Geocoding result.

See [Placemark geocode geolocation control](#).[SearchControl](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
GeocodeResult()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">IDataManager</a>	Geo object data. Inherited from <a href="#">IGeoObject</a> .
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

### Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li></ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>oldMap - Old map.</li><li>newMap - New map.</li></ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>clientX - X coordinate of the touch relative to the viewable area of the browser.</li><li>clientY - Y coordinate of the touch relative to the viewable area of the browser.</li><li>pageX - X coordinate of the touch relative to the beginning of the document.</li><li>pageY - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li><code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li><code>oldParent</code> - Old parent.</li> <li><code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getAddressLine()</a>	String	Returns string with the address of the object.
<a href="#">getAdministrativeAreas()</a>	String[]	Returns regional areas the object belongs to (federal district, region, or other district); no more than two.
<a href="#">getCountry()</a>	String null	Returns the country the toponym belongs to (if applicable).

Name	Returns	Description
<a href="#">getCountryCode()</a>	String null	Returns the code of the country the toponym belongs to (if applicable), as a two-letter ISO 3166 code.
<a href="#">getLocalities()</a>	String[]	Returns the populated locality and, optionally, an area within the locality that the toponym belongs to.
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">getPremise()</a>	String null	Returns the name of the building, if it has one (for example, the name of an airport terminal).
<a href="#">getPremiseNumber()</a>	String null	Returns the building number (including the unit, complex, or other additional characteristics).
<a href="#">getThoroughfare()</a>	String null	Returns the roadway (street, highway, road, and so on) that the toponym belongs to (if applicable).
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IChildOnMap</a> .

## Methods details

### getAddressLine

```
{String} getAddressLine()
```

**Returns** string with the address of the object.

### getAdministrativeAreas

```
{String[]} getAdministrativeAreas()
```

**Returns** regional areas the object belongs to (federal district, region, or other district); no more than two.

### getCountry

```
{String|null} getCountry()
```

**Returns** the country the toponym belongs to (if applicable).

### getCountryCode

```
{String|null} getCountryCode()
```

**Returns** the code of the country the toponym belongs to (if applicable), as a two-letter ISO 3166 code.

### getLocalities

```
{String[]} getLocalities()
```

**Returns** the populated locality and, optionally, an area within the locality that the toponym belongs to.

### getPremise

```
{String|null} getPremise()
```

**Returns** the name of the building, if it has one (for example, the name of an airport terminal).

### getPremiseNumber

```
{String|null} getPremiseNumber()
```

**Returns** the building number (including the unit, complex, or other additional characteristics).

### getThoroughfare

```
{String|null} getThoroughfare()
```

**Returns** the roadway (street, highway, road, and so on) that the toponym belongs to (if applicable).

## geolocation

Static object.

Provides information about the user's location.

[Methods](#)

**Methods**

Name	Returns	Description
<a href="#">get</a> ([options])	<a href="#">vow.Promise</a>	Tries to determine the user's location. Returns the promise object, which will either be confirmed by the object with the field <code>geoObjects</code> or rejected with an error message. The <code>geoObjects</code> field is an instance of <a href="#">GeoObjectCollection</a> . The object that indicates the user's current location will be added to the collection.

**Methods details****get**

```
{vow.Promise} get([options])
```

Tries to determine the user's location. Returns the promise object, which will either be confirmed by the object with the field `geoObjects` or rejected with an error message. The `geoObjects` field is an instance of [GeoObjectCollection](#). The object that indicates the user's current location will be added to the collection.

**Returns** Promise object.

**Parameters:**

Parameter	Default value	Description
<a href="#">options</a>	—	Type:
<a href="#">options.autoReverseGeocode</a>	true	Type:  If true, geocode the user position automatically; if false, return as it is. If automatic geocoding is used, the object marking the user's current position has the same structure as the result of executing <a href="#">geocode</a> .
<a href="#">options.mapStateAutoApply</a>	false	Type:  If true, the map center and zoom level are adjusted automatically to show the current location of the user; if false, nothing happens.

Parameter	Default value	Description
<a href="#">options.provider</a>	'auto'	Type:  Geolocation provider. Accepted values: 'yandex' - geolocation according to the Yandex data, based on the user IP-address; 'browser' - built-in browser geolocation; 'auto' - try to locate the user by all means available and then choose the best value.
<a href="#">options.timeout</a>	30000	Type:  The response time, in milliseconds.
<a href="#">options.useMapMargin</a>	true	Type: Boolean  Whether to account for map margins <a href="#">map.margin.Manager</a> when automatically centering and zooming the map.

**Examples:****1.**

```
ymaps.geolocation.get({
  // Setting the option for determining a user by IP
  provider: 'yandex',
  // The map is automatically centered by the user's position.
  mapStateAutoApply: true
}).then(function (result) {
  myMap.geoObjects.add(result.geoObjects);
});
```

**2.**

```
ymaps.geolocation.get({
  // Setting the option for detecting location by IP provider: 'yandex',
  // Automatically geocoding the result.
  autoReverseGeocode: true
}).then(function (result) {
  // Taking the data resulting from geocoding the object and outputting it to the console.
  console.log(result.geoObjects.get(0).properties.get('metaDataProperty'));
});
```

## geometry

### geometry.base

#### geometry.base.Circle

Extends [IBaseCircleGeometry](#).

The "Circle" base geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
geometry.base.Circle([coordinates[, radius]])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">coordinates</a>	null	Type: Number[] null Coordinates of the center of the circle.
<a href="#">radius</a>	0	Type: Number Radius of the circle.

**Example:**

```
var myCircle = new ymaps.geometry.base.Circle([0, 0], 10);
myCircle.events.add('change', function () {
  alert('Geometry changed');
});
myCircle.freeze();
myCircle.setCoordinates([10, 10]);
myCircle.setRadius(20);
// At this moment, a single event will be generated, and a message will be output.
myCircle.unfreeze();
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">event.Manager</a>	Geometry event manager.

**Events**

Name	Description
<a href="#">change</a>	<p>Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>oldCoordinates - Old coordinates of the center.</li> <li>newCoordinates - New coordinates of the center.</li> <li>oldRadius - Old radius.</li> <li>newRadius - New radius.</li> </ul> <p>Inherited from <a href="#">ICircleGeometryAccess</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	<p>Checks whether the passed point is located inside the circle.</p> <p>Inherited from <a href="#">ICircleGeometryAccess</a>.</p>
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	<p>Switches the object to "frozen" mode.</p> <p>Inherited from <a href="#">IFreezable</a>.</p>

Name	Returns	Description
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for a point on the circle closest to the anchorPosition.  Inherited from <a href="#">ICircleGeometryAccess</a> .
<a href="#">getCoordinates()</a>	Number[][] null	Returns coordinates of the center of the circle.  Inherited from <a href="#">ICircleGeometryAccess</a> .
<a href="#">getRadius()</a>	Number	Returns radius of the circle.  Inherited from <a href="#">ICircleGeometryAccess</a> .
<a href="#">getType()</a>	String	Returns the "Circle" string.  Inherited from <a href="#">IBaseCircleGeometry</a> .
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false.  Inherited from <a href="#">IFreezable</a> .
<a href="#">setCoordinates(coordinates)</a>	<a href="#">ICircleGeometryAccess</a>	Sets the coordinates of the center of the circle.  Inherited from <a href="#">ICircleGeometryAccess</a> .
<a href="#">setRadius(radius)</a>	<a href="#">ICircleGeometryAccess</a>	Sets the radius of the circle.  Inherited from <a href="#">ICircleGeometryAccess</a> .
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode.  Inherited from <a href="#">IFreezable</a> .

## Fields details

### events

```
{event.Manager} events
```

Geometry event manager.

## geometry.base.LinearRing

Extends [IBaseLinearRingGeometry](#).

The "Closed contour" base geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
geometry.base.LinearRing([coordinates[, fillRule]])
```

### Parameters:

Parameter	Default value	Description
<a href="#">coordinates</a>	[]	Type: Number[][]  Geometry coordinates.
<a href="#">fillRule</a>	"evenOdd"	Type: String  String ID that defines the polygon fill rule. Accepts one of two values: <ul style="list-style-type: none"> <li>• evenOdd - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it.</li> <li>• nonZero - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.</li> </ul>

### Example:

```
var linearRing = new ymaps.geometry.base.LinearRing([
  [0, 0], [0, 10], [10, 10], [10, 0], [0, 0]
]);
//...
linearRing.set(1, [5, 10]);
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">event.Manager</a>	Geometry event manager.

**Events**

Name	Description
<a href="#">change</a>	<p>Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>oldCoordinates - Old coordinates</li> <li>newCoordinates - New coordinates.</li> <li>oldFillRule - Old fill rule.</li> <li>newFillRule - New fill rule.</li> </ul> <p>Inherited from <a href="#">ILinearRingGeometryAccess</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	<p>Checks whether the passed point is located inside the contour.</p> <p>Inherited from <a href="#">ILinearRingGeometryAccess</a>.</p>
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	<p>Switches the object to "frozen" mode.</p> <p>Inherited from <a href="#">IFreezable</a>.</p>
<a href="#">get(index)</a>	Number[]	<p>Returns coordinates of the point with the specified index.</p> <p>Inherited from <a href="#">ILinearRingGeometryAccess</a>.</p>
<a href="#">getBounds()</a>	Number[][] null	<p>Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.</p> <p>Inherited from <a href="#">IBaseGeometry</a>.</p>
<a href="#">getChildGeometry(index)</a>	<a href="#">IPointGeometryAccess</a>	<p>Creates and returns an <a href="#">IPointGeometryAccess</a> object for the specified contour on the polyline.</p> <p>Inherited from <a href="#">ILinearRingGeometryAccess</a>.</p>
<a href="#">getClosest(anchorPosition)</a>	Object	<p>Searches for a point on the contour closest to the anchorPosition.</p> <p>Inherited from <a href="#">ILinearRingGeometryAccess</a>.</p>

Name	Returns	Description
<a href="#">getCoordinates()</a>	Number[][]	Returns an array of geometry coordinates.  Inherited from <a href="#">ILinearRingGeometryAccess</a> .
<a href="#">getFillRule()</a>	String	Returns ID of the fill rule.  Inherited from <a href="#">ILinearRingGeometryAccess</a> .
<a href="#">getLength()</a>	Integer	Returns the number of points in the geometry.  Inherited from <a href="#">ILinearRingGeometryAccess</a> .
<a href="#">getType()</a>	String	Returns the "LinearRing" string.  Inherited from <a href="#">IBaseLinearRingGeometry</a> .
<a href="#">insert(index, coordinates)</a>	<a href="#">ILinearRingGeometryAccess</a>	Adds a new point with the specified index.  Inherited from <a href="#">ILinearRingGeometryAccess</a> .
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false.  Inherited from <a href="#">IFreezable</a> .
<a href="#">remove(index)</a>	Number[]	Removes the point with the specified index.  Inherited from <a href="#">ILinearRingGeometryAccess</a> .
<a href="#">set(index, coordinates)</a>	<a href="#">ILinearRingGeometryAccess</a>	Sets coordinates of the point with the specified index.  Inherited from <a href="#">ILinearRingGeometryAccess</a> .
<a href="#">setCoordinates(coordinates)</a>	<a href="#">ILinearRingGeometryAccess</a>	Sets an array of geometry coordinates.  Inherited from <a href="#">ILinearRingGeometryAccess</a> .
<a href="#">setFillRule(fillRule)</a>	<a href="#">ILinearRingGeometryAccess</a>	Sets the contour fill rule.  Inherited from <a href="#">ILinearRingGeometryAccess</a> .

Name	Returns	Description
<code>splice(index, number)</code>	Number[][]	Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter.  Inherited from <a href="#">ILinearRingGeometryAccess</a> .
<code>unfreeze()</code>	<a href="#">IFreezable</a>	Switches the object to active mode.  Inherited from <a href="#">IFreezable</a> .

### Fields details

### events

```
{event.Manager} events
```

Geometry event manager.

### `geometry.base.LinearRing.fromEncodedCoordinates`

Static function.

Creates a [geometry.base.LinearRing](#) geometry based on a string of Base64-encoded coordinates.

**Returns** a geometry.

```
{ geometry.base.LinearRing } geometry.base.LinearRing.fromEncodedCoordinates(encodedCoordinates)
```

### Parameters:

Parameter	Default value	Description
<code>encodedCoordinates</code> *	—	Type: String  The Base64-encoded coordinates of contour vertexes.

\* Mandatory parameter/option.

### `geometry.base.LinearRing.toEncodedCoordinates`

Static function.

**Returns** a string of Base64-encoded coordinates for the object defined for the geometry.

```
{ String } geometry.base.LinearRing.toEncodedCoordinates(geometry)
```

### Parameters:

Parameter	Default value	Description
<code>geometry</code> *	—	Type: <a href="#">geometry.base.LinearRing</a>  Geometry.

\* Mandatory parameter/option.

**geometry.base.LineString**

Extends [IBaseLineStringGeometry](#).

The "Polyline" base geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
geometry.base.LineString([coordinates])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">coordinates</a>	[]	Type: Number[][] Geometry coordinates.

**Example:**

```
var lineString = new ymaps.geometry.base.LineString([
  [30, 50], [31, 51], [32, 52]
]);
//...
lineString.set(1, [20, 40]).remove(2);
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">event.Manager</a>	Geometry event manager.

**Events**

Name	Description
<a href="#">change</a>	<p>Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>oldCoordinates - Old coordinates</li> <li>newCoordinates - New coordinates.</li> </ul> <p>Inherited from <a href="#">ILineStringGeometryAccess</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	<p>Switches the object to "frozen" mode.</p> <p>Inherited from <a href="#">IFreezable</a>.</p>
<a href="#">get(index)</a>	Number[]	<p>Returns coordinates of the point with the specified index.</p> <p>Inherited from <a href="#">ILineStringGeometryAccess</a>.</p>

Name	Returns	Description
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getChildGeometry(index)</a>	<a href="#">IPointGeometryAccess</a>	Creates and returns an <a href="#">IPointGeometryAccess</a> object for the specified vertex on the polyline.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for a point on the polyline closest to the anchorPosition.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">getCoordinates()</a>	Number[][]	Returns an array of geometry coordinates.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">getLength()</a>	Integer	Returns the number of points in the geometry.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">getType()</a>	String	Returns the "LineString" string.  Inherited from <a href="#">IBaseLineStringGeometry</a> .
<a href="#">insert(index, coordinates)</a>	<a href="#">ILineStringGeometryAccess</a>	Adds a new point with the specified index.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false.  Inherited from <a href="#">IFreezable</a> .
<a href="#">remove(index)</a>	Number[]	Removes the point with the specified index.  Inherited from <a href="#">ILineStringGeometryAccess</a> .

Name	Returns	Description
<a href="#">set(index, coordinates)</a>	<a href="#">ILineStringGeometryAccess</a>	Sets coordinates of the point with the specified index.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">setCoordinates(coordinates)</a>	<a href="#">ILineStringGeometryAccess</a>	Sets an array of geometry coordinates.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">splice(index, number)</a>	Number[][]	Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode.  Inherited from <a href="#">IFreezable</a> .

## Fields details

### events

```
{event.Manager} events
```

Geometry event manager.

### **geometry.base.LineString.fromEncodedCoordinates**

Static function.

Creates a [geometry.base.LineString](#) geometry based on a string of Base64-encoded coordinates.

**Returns** a geometry.

```
{ geometry.base.LineString } geometry.base.LineString.fromEncodedCoordinates(encodedCoordinates)
```

### Parameters:

Parameter	Default value	Description
<a href="#">encodedCoordinates</a> *	—	Type: String  Base64-encoded coordinates of polyline points.

\* Mandatory parameter/option.

### **geometry.base.LineString.toEncodedCoordinates**

Static function.

**Returns** a string of Base64-encoded coordinates for the object defined for the geometry.

```
{ String } geometry.base.LineString.toEncodedCoordinates(geometry)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">geometry.base.LineString</a> Geometry.

\* Mandatory parameter/option.

**geometry.base.Point**

Extends [IBasePointGeometry](#).

The "Point" base geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
geometry.base.Point([coordinates])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">coordinates</a>	null	Type: <code>Number[] null</code> Coordinates of a point.

**Example:**

```
var point = new ymaps.geometry.base.Point([30, 50]);
// This point will always correspond to the map center.
map.events.add('boundschange', function (e) {
  if (e.get('newCenter') !== e.get('oldCenter')) {
    point.setCoordinates(e.get('newCenter'));
  }
});
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">event.Manager</a>	Geometry event manager.

**Events**

Name	Description
<a href="#">change</a>	<p>Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>oldCoordinates</code> - Old coordinates</li> <li><code>newCoordinates</code> - New coordinates.</li> </ul> <p>Inherited from <a href="#">IPointGeometryAccess</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getCoordinates()</a>	Number[][] null	Returns coordinates of a point.  Inherited from <a href="#">IPointGeometryAccess</a> .
<a href="#">getType()</a>	String	Returns the "Point" string.  Inherited from <a href="#">IBasePointGeometry</a> .
<a href="#">setCoordinates(coordinates)</a>	<a href="#">IPointGeometryAccess</a>	Sets coordinates of a point.  Inherited from <a href="#">IPointGeometryAccess</a> .

**Fields details****events**

```
{event.Manager} events
```

Geometry event manager.

**geometry.base.Polygon**

Extends [IBasePolygonGeometry](#).

The "Polygon" base geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
geometry.base.Polygon([coordinates[, fillRule]])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">coordinates</a>	[]	Type: Number[][]  Geometry coordinates.

Parameter	Default value	Description
<a href="#">fillRule</a>	"evenOdd"	<p>Type: String</p> <p>String ID that defines the polygon fill rule. Accepts one of two values:</p> <ul style="list-style-type: none"> <li>• <code>evenOdd</code> - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it.</li> <li>• <code>nonZero</code> - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.</li> </ul>

**Example:**

```
var polygon = new ymaps.geometry.base.Polygon([
  // Outer contour.
  [
    [0, 0], [0, 5], [5, 5], [5, 0], [0, 0]
  ],
  // Inner contour.
  [
    [1, 1], [1, 2], [2, 2], [2, 1], [1, 1]
  ]
]);
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">event.Manager</a>	Geometry event manager.

**Events**

Name	Description
<a href="#">change</a>	<p>Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>oldCoordinates</code> - Old coordinates</li> <li>• <code>newCoordinates</code> - New coordinates.</li> <li>• <code>oldFillRule</code> - Old fill rule.</li> <li>• <code>newFillRule</code> - New fill rule.</li> </ul> <p>Inherited from <a href="#">IPolygonGeometryAccess</a>.</p>

## Methods

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the polygon.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	Switches the object to "frozen" mode.  Inherited from <a href="#">IFreezable</a> .
<a href="#">get(index)</a>	Number[][]	Returns coordinates of the contour with the specified index.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getChildGeometry(index)</a>	<a href="#">ILinearRingGeometryAccess</a>	Creates and returns an <a href="#">ILinearRingGeometryAccess</a> object for the specified contour.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for the point nearest to "anchorPosition" on the polygon contour.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">getCoordinates()</a>	Number[][][]	Returns an array of geometry coordinates.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">getFillRule()</a>	String	Returns ID of the fill rule.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">getLength()</a>	Integer	Returns the number of contours in the geometry.  Inherited from <a href="#">IPolygonGeometryAccess</a> .

Name	Returns	Description
<a href="#">getType()</a>	String	Returns the "Polygon" string. Inherited from <a href="#">IBasePolygonGeometry</a> .
<a href="#">insert(index, path)</a>	<a href="#">IPolygonGeometryAccess</a>	Adds a new contour with the specified index. Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from <a href="#">IFreezable</a> .
<a href="#">remove(index)</a>	<a href="#">ILinearRingGeometryAccess</a>	Removes the contour with the specified index. Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">set(index, path)</a>	<a href="#">IPolygonGeometryAccess</a>	Sets coordinates of the contour with the specified index. Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">setCoordinates(coordinates)</a>	<a href="#">IPolygonGeometryAccess</a>	Sets an array of geometry coordinates. Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">setFillRule(fillRule)</a>	<a href="#">IPolygonGeometryAccess</a>	Sets the polygon's fill rule. Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">splice(index, number)</a>	<a href="#">ILinearRingGeometryAccess[]</a>	Deletes a defined number of contours, starting from the specified index. New contours can be added in place of the deleted ones. Coordinates of the new contours can be passed as additional arguments after the "number" parameter. Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode. Inherited from <a href="#">IFreezable</a> .

### Fields details

### events

```
{event.Manager} events
```

Geometry event manager.

**geometry.base.Polygon.fromEncodedCoordinates**

Static function.

Creates a [geometry.base.Polygon](#) geometry based on a string of Base64-encoded coordinates.

**Returns** a geometry.

```
{ geometry.base.Polygon } geometry.base.Polygon.fromEncodedCoordinates(encodedCoordinates)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">encodedCoordinates</a> *	—	Type: String  Base64-encoded coordinates of polygon points.

\* Mandatory parameter/option.

**geometry.base.Polygon.toEncodedCoordinates**

Static function.

**Returns** a string of Base64-encoded coordinates for the object defined for the geometry.

```
{ String } geometry.base.Polygon.toEncodedCoordinates(geometry)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">geometry.base.Polygon</a>  Geometry.

\* Mandatory parameter/option.

**geometry.base.Rectangle**

Extends [IBaseRectangleGeometry](#).

The "Rectangle" base geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
geometry.base.Rectangle([coordinates])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">coordinates</a>	null	Type: Number[][] null  An array containing coordinates of two opposite corners of the rectangle.

**Example:**

```
var rectangle = new ymaps.geometry.base.Rectangle([
  [30, 50], [31, 51]
]);
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">event.Manager</a>	Geometry event manager.

**Events**

Name	Description
<a href="#">change</a>	<p>Change to corner coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>oldCoordinates</code> - Old coordinates of the corners.</li> <li><code>newCoordinates</code> - New coordinates of the corners.</li> </ul> <p>Inherited from <a href="#">IRectangleGeometryAccess</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	<p>Checks whether the passed point is located inside the rectangle.</p> <p>Inherited from <a href="#">IRectangleGeometryAccess</a>.</p>
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	<p>Switches the object to "frozen" mode.</p> <p>Inherited from <a href="#">IFreezable</a>.</p>
<a href="#">getBounds()</a>	Number[][] null	<p>Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.</p> <p>Inherited from <a href="#">IBaseGeometry</a>.</p>
<a href="#">getClosest(anchorPosition)</a>	Object	<p>Searches for the point nearest to "anchorPosition" on the rectangle contour.</p> <p>Inherited from <a href="#">IRectangleGeometryAccess</a>.</p>
<a href="#">getCoordinates()</a>	Number[][]	<p>Returns coordinates of two opposite corners of the rectangle.</p> <p>Inherited from <a href="#">IRectangleGeometryAccess</a>.</p>
<a href="#">getType()</a>	String	<p>Returns the "Rectangle" string.</p> <p>Inherited from <a href="#">IBaseRectangleGeometry</a>.</p>

Name	Returns	Description
<code>isFrozen()</code>	Boolean	Returns true if the object is in "frozen" mode, otherwise false.  Inherited from <a href="#">IFreezable</a> .
<code>setCoordinates(coordinates)</code>	<a href="#">IRectangleGeometryAccess</a>	Sets the coordinates of two opposite corners of the rectangle.  Inherited from <a href="#">IRectangleGeometryAccess</a> .
<code>unfreeze()</code>	<a href="#">IFreezable</a>	Switches the object to active mode.  Inherited from <a href="#">IFreezable</a> .

### Fields details

#### events

```
{event.Manager} events
```

Geometry event manager.

## geometry.Circle

Extends [ICircleGeometry](#).

The "Circle" geometry.

See [Circle](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
geometry.Circle([coordinates[, radius[, options]])
```

#### Parameters:

Parameter	Default value	Description
<code>coordinates</code>	null	Type: Number[] null  Coordinates of the center of the circle.
<code>radius</code>	0	Type: Number  Radius of the circle in meters.
<code>options</code>	—	Type: Object  Geometry options.
<code>options.geodesic</code>	false	Type: Boolean  Enables display using geodesic lines.

Parameter	Default value	Description
<a href="#">options.pixelRendering</a>	"jumpy"	Type: String  Method for calculating pixel coordinates of the shape in cycled projections. This option accepts one of the following values: <ul style="list-style-type: none"> <li>jumpy - The shape is placed as close as possible to the center of the map viewport, and can "jump" when the map is being moved.</li> <li>static - The shape is always located in the initial world and does not move when the map is moved.</li> </ul>
<a href="#">options.projection</a>	—	Type: <a href="#">IProjection</a>  Projection.

**Example:**

```
// Creating an instance of the circle geometry class (specifying the coordinates and radius in meters).
var circleGeometry = new ymaps.geometry.Circle([30, 50], 10);
// Creating an instance of the geo object class and passing our geometry to the constructor.
var circleGeoObject = new ymaps.GeoObject({ geometry: circleGeometry });

// Changing the geometry's radius via the geo object's "geometry" property.
circleGeoObject.geometry.setRadius(5)
// Or directly.
circleGeometry.setRadius(5);
// You can also access circleGeometry via circleGeoObject.geometry.
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager.  Inherited from <a href="#">ICustomizable</a> .

**Events**

Name	Description
<a href="#">change</a>	Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>oldCoordinates - Old coordinates of the center.</li> <li>newCoordinates - New coordinates of the center.</li> <li>oldRadius - Old radius.</li> <li>newRadius - New radius.</li> </ul> Inherited from <a href="#">ICircleGeometryAccess</a> .
<a href="#">mapchange</a>	Map reference changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> Inherited from <a href="#">IGeometry</a> .

Name	Description
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">pixelgeometrychange</a>	The pixel geometry changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>pixelGeometry</code> - New <a href="#">IPixelGeometry</a> pixel geometry.</li> </ul> Inherited from <a href="#">IGeometry</a> .

## Methods

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the circle.  Inherited from <a href="#">ICircleGeometryAccess</a> .
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	Switches the object to "frozen" mode.  Inherited from <a href="#">IFreezable</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for a point on the circle closest to the anchorPosition.  Inherited from <a href="#">ICircleGeometryAccess</a> .
<a href="#">getCoordinates()</a>	Number[] null	Returns coordinates of the center of the circle.  Inherited from <a href="#">ICircleGeometryAccess</a> .
<a href="#">getMap()</a>	<a href="#">Map</a>  null	Returns the current map.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getPixelGeometry([options])</a>	<a href="#">IPixelGeometry</a>	Returns the pixel geometry corresponding to the given geometry, its options, and the map state.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getRadius()</a>	Number	Returns radius of the circle.  Inherited from <a href="#">ICircleGeometryAccess</a> .

Name	Returns	Description
<a href="#">getType()</a>	String	Returns the "Circle" string. Inherited from <a href="#">ICircleGeometry</a> .
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from <a href="#">IFreezable</a> .
<a href="#">setCoordinates(coordinates)</a>	<a href="#">ICircleGeometryAccess</a>	Sets the coordinates of the center of the circle. Inherited from <a href="#">ICircleGeometryAccess</a> .
<a href="#">setMap(map)</a>		Sets the map. Inherited from <a href="#">IGeometry</a> .
<a href="#">setRadius(radius)</a>	<a href="#">ICircleGeometryAccess</a>	Sets the radius of the circle. Inherited from <a href="#">ICircleGeometryAccess</a> .
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode. Inherited from <a href="#">IFreezable</a> .

## geometry.json

### geometry.json.circle

Extends [IGeometryJson](#).

An object that defines the JSON representation of the "Circle" geometry.

[Constructor](#) | [Fields](#)

### Constructor

```
geometry.json.circle()
```

### Example:

```
var geometryJson = {
  type: "Circle",
  coordinates: [1, 2],
  radius: 100
};
```

### Fields

Name	Type	Description
<a href="#">coordinates</a>	Number[] null	Coordinates of the center of the circle.
<a href="#">radius</a>	Number	Radius of the circle.
<a href="#">type</a>	String	ID of the "Circle" geometry type. It must always take the value "Circle".

**Fields details****coordinates**

```
{Number[]|null} coordinates
```

Coordinates of the center of the circle.

**radius**

```
{Number} radius
```

Radius of the circle.

**type**

```
{String} type
```

ID of the "Circle" geometry type. It must always take the value "Circle".

**geometry.json.lineString**

Extends [IGeometryJson](#).

An object describing the JSON representation of the "Polyline" geometry.

[Constructor](#) | [Fields](#)

**Constructor**

```
geometry.json.lineString()
```

**Example:**

```
var geometryJson = {
  type: "LineString",
  coordinates: [[1, 2], [3, 5], [7, 11]]
};
```

**Fields**

Name	Type	Description
<a href="#">coordinates</a>	Number[][]	Coordinates of a polyline.
<a href="#">type</a>	String	Identifier of the "Polyline" geometry type. Must always take the value "LineString".

**Fields details****coordinates**

```
{Number[][]} coordinates
```

Coordinates of a polyline.

**type**

```
{String} type
```

Identifier of the "Polyline" geometry type. Must always take the value "LineString".

## geometry.json.Point

Extends [IGeometryJson](#).

An object that defines the JSON representation of the "Point" geometry.

[Constructor](#) | [Fields](#)

### Constructor

```
geometry.json.Point()
```

### Example:

```
var geometryJson = {  
  type: "Point",  
  coordinates: [1, 2]  
};
```

### Fields

Name	Type	Description
<a href="#">type</a>	String	ID of the geometry type. Inherited from <a href="#">IGeometryJson</a> .

## geometry.json.polygon

Extends [IGeometryJson](#).

An object that defines the JSON representation of the "Polygon" geometry.

[Constructor](#) | [Fields](#)

### Constructor

```
geometry.json.polygon()
```

### Example:

```
var geometryJson = {  
  type: "Polygon",  
  coordinates: [  
    [[0, 0], [7, 11]],  
    [[1, 2], [3, 5]]  
  ]  
};
```

### Fields

Name	Type	Description
<a href="#">coordinates</a>	Number[][][]	Coordinates of the polygon.

Name	Type	Description
<a href="#">fillRule</a>	String	ID of the polygon fill rule. Accepts one of two values: <ul style="list-style-type: none"><li>• <code>evenOdd</code> - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it.</li><li>• <code>nonZero</code> - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.</li></ul>
<a href="#">type</a>	String	ID of the geometry type. Inherited from <a href="#">IGeometryJson</a> .

## Fields details

### coordinates

```
{Number[][][]} coordinates
```

Coordinates of the polygon.

### fillRule

```
{String} fillRule
```

ID of the polygon fill rule. Accepts one of two values:

- `evenOdd` - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it.
- `nonZero` - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.

### geometry.json.rectangle

Extends [IGeometryJson](#).

An object that defines the JSON representation of the "Rectangle" geometry.

[Constructor](#) | [Fields](#)

### Constructor

```
geometry.json.rectangle()
```

### Example:

```
var geometryJson = {  
  type: "Rectangle",  
  coordinates: [[1, 2], [3, 5]]  
};
```

**Fields**

Name	Type	Description
<a href="#">coordinates</a>	Number[][] null	Coordinates of two opposite corners of the rectangle.
<a href="#">type</a>	String	ID of the "Rectangle" geometry type. It must always take the value "Rectangle".

**Fields details****coordinates**

```
{Number[][]|null} coordinates
```

Coordinates of two opposite corners of the rectangle.

**type**

```
{String} type
```

ID of the "Rectangle" geometry type. It must always take the value "Rectangle".

**geometry.LineString**

Extends [ILineStringGeometry](#).

"Polyline" geometry.

See [Polyline](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
geometry.LineString([coordinates[, options]])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">coordinates</a>	[]	Type: Number[][]  Geometry coordinates.
<a href="#">options</a>	—	Type: Object  Geometry options.
<a href="#">options.coordRendering</a>	—	Type: String  String ID defining the algorithm for recalculating geometry coordinates as pixel coordinates. For the "Polyline" geometry, can accept one of two values: <ul style="list-style-type: none"> <li>shortestPath - Algorithm that considers projection cycling on the axes and generates pixel coordinates so that the distance between two neighboring points is minimal.</li> <li>straightPath - Algorithm that does not consider projection cycling.</li> </ul>

Parameter	Default value	Description
<a href="#">options.geodesic</a>	false	Type: Boolean  Enables display using geodesic lines.
<a href="#">options.pixelRendering</a>	"jumpy"	Type: String  Method for calculating pixel coordinates of the shape in cycled projections. This option accepts one of the following values: <ul style="list-style-type: none"> <li>jumpy - The shape is placed as close as possible to the center of the map viewport, and can "jump" when the map is being moved.</li> <li>static - The shape is always located in the initial world and does not move when the map is moved.</li> </ul>
<a href="#">options.projection</a>	—	Type: <a href="#">IProjection</a>  Projection.
<a href="#">options.simplification</a>	true	Type: Boolean  Enables simplification during rendering of a pixel geometry.

**Example:**

```
// Instantiates the point geometry (specifying coordinates).
var lineStringGeometry = new ymaps.geometry.LineString([
  [30, 50], [31, 51], [32, 52]
]);
// Instantiating the geo object and passing our geometry to the constructor.
var lineStringGeoObject = new ymaps.GeoObject({ geometry: lineStringGeometry });

lineStringGeometry.events.add('change', function (e) {
  alert([e.get('newCoordinates'), e.get('oldCoordinates')]);
});

// Changing vertexes via the geo object's "geometry" property (setting new coordinates for the second point on the line).
lineStringGeoObject.geometry
  .set(1, [20, 40])
  .remove(2);
// Or directly.
lineStringGeometry
  .set(1, [20, 40])
  .remove(2);
// You can also access lineStringGeometry via lineStringGeoObject.geometry.
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager.  Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">change</a>	<p>Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>oldCoordinates - Old coordinates</li> <li>newCoordinates - New coordinates.</li> </ul> <p>Inherited from <a href="#">ILineStringGeometryAccess</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> <p>Inherited from <a href="#">IGeometry</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">pixelgeometrychange</a>	<p>The pixel geometry changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>pixelGeometry - New <a href="#">IPixelGeometry</a> pixel geometry.</li> </ul> <p>Inherited from <a href="#">IGeometry</a>.</p>

## Methods

Name	Returns	Description
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	<p>Switches the object to "frozen" mode.</p> <p>Inherited from <a href="#">IFreezable</a>.</p>
<a href="#">get(index)</a>	Number[]	<p>Returns coordinates of the point with the specified index.</p> <p>Inherited from <a href="#">ILineStringGeometryAccess</a>.</p>
<a href="#">getBounds()</a>	Number[][] null	<p>Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner.</p> <p>Inherited from <a href="#">IGeometry</a>.</p>
<a href="#">getChildGeometry(index)</a>	<a href="#">IPointGeometryAccess</a>	<p>Creates and returns an <a href="#">IPointGeometryAccess</a> object for the specified vertex on the polyline.</p> <p>Inherited from <a href="#">ILineStringGeometryAccess</a>.</p>
<a href="#">getClosest(anchorPosition)</a>	Object	<p>Searches for a point on the polyline closest to the anchorPosition.</p> <p>Inherited from <a href="#">ILineStringGeometryAccess</a>.</p>

Name	Returns	Description
<a href="#">getCoordinates()</a>	Number[][]	Returns an array of geometry coordinates.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">getDistance([from[, to]])</a>	Number	Returns the length of the specified line segment, or the whole line, if the delimiter is not set.
<a href="#">getLength()</a>	Integer	Returns the number of points in the geometry.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">getMap()</a>	Map null	Returns the current map.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getPixelGeometry([options])</a>	IPixelGeometry	Returns the pixel geometry corresponding to the given geometry, its options, and the map state.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getType()</a>	String	Returns the "LineString" string.  Inherited from <a href="#">ILineStringGeometry</a> .
<a href="#">insert(index, coordinates)</a>	ILineStringGeometryAccess	Adds a new point with the specified index.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false.  Inherited from <a href="#">IFreezable</a> .
<a href="#">remove(index)</a>	Number[]	Removes the point with the specified index.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">set(index, coordinates)</a>	ILineStringGeometryAccess	Sets coordinates of the point with the specified index.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">setCoordinates(coordinates)</a>	ILineStringGeometryAccess	Sets an array of geometry coordinates.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">setMap(map)</a>		Sets the map.  Inherited from <a href="#">IGeometry</a> .

Name	Returns	Description
<code>splice(index, number)</code>	Number[][]	<p>Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter.</p> <p>Inherited from <a href="#">ILineStringGeometryAccess</a>.</p>
<code>unfreeze()</code>	<a href="#">IFreezable</a>	<p>Switches the object to active mode.</p> <p>Inherited from <a href="#">IFreezable</a>.</p>

## Methods details

### getDistance

```
{Number} getDistance([from[, to]])
```

**Returns** the length of the specified line segment, or the whole line, if the delimiter is not set.

#### Parameters:

Parameter	Default value	Description
<code>from</code>	0	<p>Type: Number</p> <p>Specifies the start point for calculating the length.</p>
<code>to</code>	—	<p>Type: Number</p> <p>Specifies the end point for calculating the length. If not specified, the last point is used.</p>

#### Example:

```
var lineStringGeometry = new ymaps.geometry.LineString([[30, 50], [31, 51], [32, 52]]);
var geoObject = new ymaps.GeoObject({ geometry: lineStringGeometry });
myMap.geoObjects.add(geoObject);
// The total length of the line.
console.log(geoObject.geometry.getDistance());
// The length of the segment from the first to the second point.
console.log(geoObject.geometry.getDistance(0, 1));
```

### geometry.LineString.fromEncodedCoordinates

Static function.

Creates a [geometry.LineString](#) geometry based on a string of Base64-encoded coordinates.

**Returns** a geometry.

```
{ geometry.LineString } geometry.LineString.fromEncodedCoordinates(encodedCoordinates)
```

#### Parameters:

Parameter	Default value	Description
<a href="#">encodedCoordinates</a> *	—	Type: String  Base64-encoded coordinates of polyline points.

\* Mandatory parameter/option.

#### Example:

```
var base64Coords = "gMPJAYDw-gJAQg8AQEIPAEBDwBAQg8A";
var geometry = ymaps.geometry.LineString.fromEncodedCoordinates(base64Coords);
var polyline = new ymaps.Polyline(geometry);
```

### geometry.LineString.toEncodedCoordinates

Static function.

**Returns** a string of Base64-encoded coordinates for the object defined for the geometry.

```
{ String } geometry.LineString.toEncodedCoordinates(geometry)
```

#### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">geometry.LineString</a>  Geometry.

\* Mandatory parameter/option.

## geometry.pixel

### geometry.pixel.Circle

Extends [IPixelCircleGeometry](#).

The "Circle" pixel geometry.

[Constructor](#) | [Fields](#) | [Methods](#)

#### Constructor

```
geometry.pixel.Circle(coordinates, radius[, metaData])
```

#### Parameters:

Parameter	Default value	Description
<a href="#">coordinates</a> *	—	Type: Number[] null  Coordinates of the center of the circle.
<a href="#">radius</a> *	—	Type: Number null  Radius of the circle.
<a href="#">metaData</a>	—	Type: Object  Metadata.

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

**Methods**

Name	Returns	Description
<a href="#">equals(geometry)</a>	Boolean	Returns true if the passed geometry is equivalent to the given one.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getCoordinates()</a>	Number[]	Returns coordinates of the center of the circle.  Inherited from <a href="#">IPixelCircleGeometry</a> .
<a href="#">getMetaData()</a>	Object	Returns metadata of the pixel geometry.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getRadius()</a>	Number	Returns radius of the circle.  Inherited from <a href="#">IPixelCircleGeometry</a> .
<a href="#">getType()</a>	String	Returns ID of the geometry type.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">scale(factor)</a>	<a href="#">IPixelGeometry</a>	Creates a scaled copy of the geometry.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">shift(offset)</a>	<a href="#">IPixelGeometry</a>	Creates a copy of the geometry that is shifted by the specified amount.  Inherited from <a href="#">IPixelGeometry</a> .

**geometry.pixel.LineString**

Extends [IPixelLineStringGeometry](#).

The "Polyline" pixel geometry.

[Constructor](#) | [Fields](#) | [Methods](#)

**Constructor**

```
geometry.pixel.LineString(coordinates[, metaData])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">coordinates</a> *	—	Type: Number[][] Coordinates of a line.
<a href="#">metaData</a>	—	Type: Object Metadata.

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

**Methods**

Name	Returns	Description
<a href="#">equals(geometry)</a>	Boolean	Returns true if the passed geometry is equivalent to the given one.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for a point on the polyline closest to the anchorPosition.  Inherited from <a href="#">IPixelLineStringGeometry</a> .

Name	Returns	Description
<a href="#">getCoordinates()</a>	Number[][]	Returns coordinates of a line. Inherited from <a href="#">IPixelLineStringGeometry</a> .
<a href="#">getLength()</a>	Integer	Returns the number of points in the geometry. Inherited from <a href="#">IPixelLineStringGeometry</a> .
<a href="#">getMetaData()</a>	Object	Returns metadata of the pixel geometry. Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getType()</a>	String	Returns ID of the geometry type. Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">scale(factor)</a>	<a href="#">IPixelGeometry</a>	Creates a scaled copy of the geometry. Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">shift(offset)</a>	<a href="#">IPixelGeometry</a>	Creates a copy of the geometry that is shifted by the specified amount. Inherited from <a href="#">IPixelGeometry</a> .

### geometry.pixel.MultiLineString

Extends [IPixelMultiLineGeometry](#).

The "Multiline" pixel geometry.

[Constructor](#) | [Fields](#) | [Methods](#)

#### Constructor

```
geometry.pixel.MultiLineString(coordinates[, metaData])
```

#### Parameters:

Parameter	Default value	Description
<a href="#">coordinates</a> *	—	Type: Number[][] Line coordinates.
<a href="#">metaData</a>	—	Type: Object Metadata.

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

**Methods**

Name	Returns	Description
<a href="#">equals(geometry)</a>	Boolean	Returns true if the passed geometry is equivalent to the given one.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for a point on the contour closest to the anchorPosition.  Inherited from <a href="#">IPixelMultiLineGeometry</a> .
<a href="#">getCoordinates()</a>	Number[][][]	Returns coordinates of a multiline.  Inherited from <a href="#">IPixelMultiLineGeometry</a> .
<a href="#">getLength()</a>	Integer	Returns the number of lines in the multiline.  Inherited from <a href="#">IPixelMultiLineGeometry</a> .
<a href="#">getMetaData()</a>	Object	Returns metadata of the pixel geometry.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getType()</a>	String	Returns ID of the geometry type.  Inherited from <a href="#">IBaseGeometry</a> .

Name	Returns	Description
<a href="#">scale(factor)</a>	<a href="#">IPixelGeometry</a>	Creates a scaled copy of the geometry.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">shift(offset)</a>	<a href="#">IPixelGeometry</a>	Creates a copy of the geometry that is shifted by the specified amount.  Inherited from <a href="#">IPixelGeometry</a> .

### geometry.pixel.MultiPolygon

Extends [IPixelMultiPolygonGeometry](#).

The "Polygon from multiple shapes" pixel geometry.

[Constructor](#) | [Fields](#) | [Methods](#)

#### Constructor

```
geometry.pixel.MultiPolygon(coordinates, fillRule[, metaData])
```

#### Parameters:

Parameter	Default value	Description
<a href="#">coordinates</a> *	—	Type: Number[]  Coordinates of the polygons.
<a href="#">fillRule</a> *	—	Type: String  String ID that defines the fill rule for the polygons. Accepts one of two values: <ul style="list-style-type: none"> <li>evenOdd - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it.</li> <li>nonZero - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.</li> </ul>

Parameter	Default value	Description
<a href="#">metaData</a>	—	Type: Object Metadata.
<a href="#">metaData.convex</a>	false	Type: Boolean Convex indicator for a polygon. If true, it is convex; if false, it is not. For convex polygons, it is faster to calculate whether points fall in the polygon.

\* Mandatory parameter/option.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

### Methods

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the multipolygon. Inherited from <a href="#">IPixelMultiPolygonGeometry</a> .
<a href="#">equals(geometry)</a>	Boolean	Returns true if the passed geometry is equivalent to the given one. Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for the point nearest to "anchorPosition" on the multipolygon contour. Inherited from <a href="#">IPixelMultiPolygonGeometry</a> .

Name	Returns	Description
<a href="#">getCoordinates()</a>	Number[][][]	Returns coordinates of the multipolygon.  Inherited from <a href="#">IPixelMultiPolygonGeometry</a> .
<a href="#">getFillRule()</a>	String	Returns the string ID that defines the multipolygon fill rule. The ID can have one of two values: <ul style="list-style-type: none"><li>• evenOdd - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it.</li><li>• nonZero - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.</li></ul> Inherited from <a href="#">IPixelMultiPolygonGeometry</a> .
<a href="#">getLength()</a>	Integer	Returns the number of polygons in the multipolygon.  Inherited from <a href="#">IPixelMultiPolygonGeometry</a> .
<a href="#">getMetaData()</a>	Object	Returns metadata of the pixel geometry.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getType()</a>	String	Returns ID of the geometry type.  Inherited from <a href="#">IBaseGeometry</a> .

Name	Returns	Description
<a href="#">scale(factor)</a>	<a href="#">IPixelGeometry</a>	Creates a scaled copy of the geometry.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">shift(offset)</a>	<a href="#">IPixelGeometry</a>	Creates a copy of the geometry that is shifted by the specified amount.  Inherited from <a href="#">IPixelGeometry</a> .

### geometry.pixel.Point

Extends [IPixelPointGeometry](#).

The "Point" pixel geometry.

[Constructor](#) | [Fields](#) | [Methods](#)

#### Constructor

```
geometry.pixel.Point(position[, metaData])
```

#### Parameters:

Parameter	Default value	Description
<a href="#">position</a> *	—	Type: Number[] null  Coordinates of a point.
<a href="#">metaData</a>	—	Type: Object  Metadata.

\* Mandatory parameter/option.

#### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

#### Methods

Name	Returns	Description
<a href="#">equals(geometry)</a>	Boolean	Returns true if the passed geometry is equivalent to the given one.  Inherited from <a href="#">IPixelGeometry</a> .

Name	Returns	Description
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getCoordinates()</a>	Number[]	Returns coordinates of a point.  Inherited from <a href="#">IPixelPointGeometry</a> .
<a href="#">getMetaData()</a>	Object	Returns metadata of the pixel geometry.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getType()</a>	String	Returns ID of the geometry type.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">scale(factor)</a>	<a href="#">IPixelGeometry</a>	Creates a scaled copy of the geometry.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">shift(offset)</a>	<a href="#">IPixelGeometry</a>	Creates a copy of the geometry that is shifted by the specified amount.  Inherited from <a href="#">IPixelGeometry</a> .

### geometry.pixel.Polygon

Extends [IPixelPolygonGeometry](#).

The "Polygon" pixel geometry.

[Constructor](#) | [Fields](#) | [Methods](#)

#### Constructor

```
geometry.pixel.Polygon(coordinates, fillRule[, metaData])
```

#### Parameters:

Parameter	Default value	Description
<a href="#">coordinates</a> *	—	Type: Number[][]  Coordinates of the polygon.

Parameter	Default value	Description
<a href="#">fillRule</a> *	—	<p>Type: String</p> <p>String ID that defines the polygon fill rule. Accepts one of two values:</p> <ul style="list-style-type: none"> <li>• <code>evenOdd</code> - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it.</li> <li>• <code>nonZero</code> - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.</li> </ul>
<a href="#">metaData</a>	—	<p>Type: Object</p> <p>Metadata.</p>
<a href="#">metaData.convex</a>	false	<p>Type: Boolean</p> <p>Convex indicator for a polygon. If true, it is convex; if false, it is not. For convex polygons, it is faster to calculate whether points fall in the polygon.</p>

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	<p>Event manager.</p> <p>Inherited from <a href="#">IEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	<p>Checks whether the passed point is located inside the polygon.</p> <p>Inherited from <a href="#">IPixelPolygonGeometry</a>.</p>

Name	Returns	Description
<a href="#">equals(geometry)</a>	Boolean	Returns true if the passed geometry is equivalent to the given one.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for the point nearest to "anchorPosition" on the polygon contour.  Inherited from <a href="#">IPixelPolygonGeometry</a> .
<a href="#">getCoordinates()</a>	Number[][]	Returns coordinates of the polygon.  Inherited from <a href="#">IPixelPolygonGeometry</a> .

Name	Returns	Description
<a href="#">getFillRule()</a>	String	<p>Returns string ID that defines the polygon fill rule. The ID accepts one of two values:</p> <ul style="list-style-type: none"><li>• <code>evenOdd</code> - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it.</li><li>• <code>nonZero</code> - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.</li></ul> <p>Inherited from <a href="#">IPixelPolygonGeometry</a>.</p>
<a href="#">getLength()</a>	Integer	<p>Returns the number of contours in the polygon.</p> <p>Inherited from <a href="#">IPixelPolygonGeometry</a>.</p>
<a href="#">getMetaData()</a>	Object	<p>Returns metadata of the pixel geometry.</p> <p>Inherited from <a href="#">IPixelGeometry</a>.</p>
<a href="#">getType()</a>	String	<p>Returns ID of the geometry type.</p> <p>Inherited from <a href="#">IBaseGeometry</a>.</p>
<a href="#">scale(factor)</a>	<a href="#">IPixelGeometry</a>	<p>Creates a scaled copy of the geometry.</p> <p>Inherited from <a href="#">IPixelGeometry</a>.</p>

Name	Returns	Description
<a href="#">shift(offset)</a>	<a href="#">IPixelGeometry</a>	Creates a copy of the geometry that is shifted by the specified amount.  Inherited from <a href="#">IPixelGeometry</a> .

### geometry.pixel.Rectangle

Extends [IPixelRectangleGeometry](#).

The "Rectangle" pixel geometry.

[Constructor](#) | [Fields](#) | [Methods](#)

#### Constructor

```
geometry.pixel.Rectangle([coordinates[, metaData]])
```

#### Parameters:

Parameter	Default value	Description
<a href="#">coordinates</a>	null	Type: Number[][] null  Coordinates of two opposite corners of the rectangle.
<a href="#">metaData</a>	—	Type: Object  Metadata.

#### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

#### Methods

Name	Returns	Description
<a href="#">equals(geometry)</a>	Boolean	Returns true if the passed geometry is equivalent to the given one.  Inherited from <a href="#">IPixelGeometry</a> .

Name	Returns	Description
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for the point nearest to "anchorPosition" on the rectangle.  Inherited from <a href="#">IPixelRectangleGeometry</a> .
<a href="#">getCoordinates()</a>	Number[][]	Returns coordinates of two opposite corners of the rectangle.  Inherited from <a href="#">IPixelRectangleGeometry</a> .
<a href="#">getMetaData()</a>	Object	Returns metadata of the pixel geometry.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getType()</a>	String	Returns ID of the geometry type.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">scale(factor)</a>	<a href="#">IPixelGeometry</a>	Creates a scaled copy of the geometry.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">shift(offset)</a>	<a href="#">IPixelGeometry</a>	Creates a copy of the geometry that is shifted by the specified amount.  Inherited from <a href="#">IPixelGeometry</a> .

## geometry.Point

Extends [IPointGeometry](#).

The "Point" geometry.

See [Placemark](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
geometry.Point([position[, options]])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">position</a>	null	Type: Number[]  Coordinates of a point.
<a href="#">options</a>	—	Type: Object  Geometry options.
<a href="#">options.pixelRendering</a>	"jumpy"	Type: String  Method for calculating pixel coordinates of the shape in cycled projections. This option accepts one of the following values: <ul style="list-style-type: none"> <li>jumpy - The shape is placed as close as possible to the center of the map viewport, and can "jump" when the map is being moved.</li> <li>static - The shape is always located in the initial world and does not move when the map is moved.</li> </ul>
<a href="#">options.projection</a>	—	Type: <a href="#">IProjection</a>  Projection.

**Example:**

```
// Creating an instance of the point geometry (specifying coordinates).
var pointGeometry = new ymaps.geometry.Point([30, 50]);
// Instantiating the geo object and passing our geometry to the constructor.
var placemark = new ymaps.GeoObject({ geometry: pointGeometry });

// Changing the vertexes via the geo object's "geometry" property.
placemark.geometry.setCoordinates([20, 40]);
// Or directly.
pointGeometry.setCoordinates([20, 40]);
// You can also access pointGeometry via placemark.geometry
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager.  Inherited from <a href="#">ICustomizable</a> .

**Events**

Name	Description
<a href="#">change</a>	Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>oldCoordinates - Old coordinates</li> <li>newCoordinates - New coordinates.</li> </ul> Inherited from <a href="#">IPointGeometryAccess</a> .

Name	Description
<a href="#">mapchange</a>	Map reference changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> Inherited from <a href="#">IGeometry</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">pixelgeometrychange</a>	The pixel geometry changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>pixelGeometry - New <a href="#">IPixelGeometry</a> pixel geometry.</li> </ul> Inherited from <a href="#">IGeometry</a> .

## Methods

Name	Returns	Description
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getCoordinates()</a>	Number[] null	Returns coordinates of a point.  Inherited from <a href="#">IPointGeometryAccess</a> .
<a href="#">getMap()</a>	Map null	Returns the current map.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getPixelGeometry([options])</a>	<a href="#">IPixelGeometry</a>	Returns the pixel geometry corresponding to the given geometry, its options, and the map state.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getType()</a>	String	Returns the "Point" string.  Inherited from <a href="#">IPointGeometry</a> .
<a href="#">setCoordinates(coordinates)</a>	<a href="#">IPointGeometryAccess</a>	Sets coordinates of a point.  Inherited from <a href="#">IPointGeometryAccess</a> .
<a href="#">setMap(map)</a>		Sets the map.  Inherited from <a href="#">IGeometry</a> .

## geometry.Polygon

Extends [IPolygonGeometry](#).

The "Polygon" geometry.

See [Polygon](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
geometry.Polygon([coordinates[, fillRule[, options]])
```

### Parameters:

Parameter	Default value	Description
<a href="#">coordinates</a>	[]	Type: Number[][][]  Geometry coordinates. A three-dimensional array of elements that are each two-dimensional coordinates of the polygon contours. The first element describes the outer contour, and the rest describe the inner contours.
<a href="#">fillRule</a>	"evenOdd"	Type: String  String ID that defines the polygon fill rule. Accepts one of two values: <ul style="list-style-type: none"><li>• evenOdd - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it.</li><li>• nonZero - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.</li></ul>
<a href="#">options</a>	—	Type: Object  Geometry options.

Parameter	Default value	Description
<a href="#">options.coordRendering</a>	"shortestPath"	Type: String  String ID defining the algorithm for recalculating geometry coordinates as pixel coordinates. Accepts one of two values: <ul style="list-style-type: none"> <li>shortestPath - Algorithm that considers projection cycling on the axes and generates pixel coordinates so that the distance between two neighboring points is minimal.</li> <li>straightPath - Algorithm that does not consider projection cycling.</li> </ul>
<a href="#">options.geodesic</a>	false	Type: Boolean  Enables display using geodesic lines.
<a href="#">options.pixelRendering</a>	"jumpy"	Type: String  Method for calculating pixel coordinates of the shape in cycled projections. This option accepts one of the following values: <ul style="list-style-type: none"> <li>jumpy - The shape is placed as close as possible to the center of the map viewport, and can "jump" when the map is being moved.</li> <li>static - The shape is always located in the initial world and does not move when the map is moved.</li> </ul>
<a href="#">options.projection</a>	—	Type: <a href="#">IProjection</a>  Projection.
<a href="#">options.simplification</a>	true	Type: Boolean  Enables simplification during rendering of a pixel geometry.

**Example:**

```
// Creating an instance of the polygon geometry (specifying coordinates of vertexes on contours).
var polygonGeometry = new ymaps.geometry.Polygon([
  // Outer contour.
  [
    [0, 0], [0, 5], [5, 5], [5, 0], [0, 0]
  ],
  // Inner contour.
  [
    [1, 1], [1, 2], [2, 2], [2, 1], [1, 1]
  ]
]);
// Creating a geo object instance and passing our geometry to the constructor.
var polygonGeoObject = new ymaps.GeoObject({ geometry: polygonGeometry });

// You can also access polygonGeometry via polygonGeoObject.geometry.
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

**Events**

Name	Description
<a href="#">change</a>	Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>oldCoordinates - Old coordinates</li> <li>newCoordinates - New coordinates.</li> <li>oldFillRule - Old fill rule.</li> <li>newFillRule - New fill rule.</li> </ul> Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">mapchange</a>	Map reference changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> Inherited from <a href="#">IGeometry</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">pixelgeometrychange</a>	The pixel geometry changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>pixelGeometry - New <a href="#">IPixelGeometry</a> pixel geometry.</li> </ul> Inherited from <a href="#">IGeometry</a> .

**Methods**

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the polygon.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	Switches the object to "frozen" mode.  Inherited from <a href="#">IFreezable</a> .
<a href="#">get(index)</a>	Number[][]	Returns coordinates of the contour with the specified index.  Inherited from <a href="#">IPolygonGeometryAccess</a> .

Name	Returns	Description
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getChildGeometry(index)</a>	<a href="#">ILinearRingGeometryAccess</a>	Creates and returns an <a href="#">ILinearRingGeometryAccess</a> object for the specified contour.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for the point nearest to "anchorPosition" on the polygon contour.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">getCoordinates()</a>	Number[][][]	Returns an array of geometry coordinates.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">getFillRule()</a>	String	Returns ID of the fill rule.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">getLength()</a>	Integer	Returns the number of contours in the geometry.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">getMap()</a>	<a href="#">Map</a>  null	Returns the current map.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getPixelGeometry([options])</a>	<a href="#">IPixelGeometry</a>	Returns the pixel geometry corresponding to the given geometry, its options, and the map state.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getType()</a>	String	Returns the "Polygon" string.  Inherited from <a href="#">IPolygonGeometry</a> .
<a href="#">insert(index, path)</a>	<a href="#">IPolygonGeometryAccess</a>	Adds a new contour with the specified index.  Inherited from <a href="#">IPolygonGeometryAccess</a> .

Name	Returns	Description
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false.  Inherited from <a href="#">IFreezable</a> .
<a href="#">remove(index)</a>	<a href="#">ILinearRingGeometryAccess</a>	Removes the contour with the specified index.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">set(index, path)</a>	<a href="#">IPolygonGeometryAccess</a>	Sets coordinates of the contour with the specified index.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">setCoordinates(coordinates)</a>	<a href="#">IPolygonGeometryAccess</a>	Sets an array of geometry coordinates.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">setFillRule(fillRule)</a>	<a href="#">IPolygonGeometryAccess</a>	Sets the polygon's fill rule.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">setMap(map)</a>		Sets the map.  Inherited from <a href="#">IGeometry</a> .
<a href="#">splice(index, number)</a>	<a href="#">ILinearRingGeometryAccess</a> []	Deletes a defined number of contours, starting from the specified index. New contours can be added in place of the deleted ones. Coordinates of the new contours can be passed as additional arguments after the "number" parameter.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode.  Inherited from <a href="#">IFreezable</a> .

### **geometry.Polygon.fromEncodedCoordinates**

Static function.

Creates a [geometry.Polygon](#) geometry based on a string of Base64-encoded coordinates.

**Returns** a geometry.

```
{ geometry.Polygon } geometry.Polygon.fromEncodedCoordinates(encodedCoordinates)
```

**Parameters:**



Parameter	Default value	Description
<a href="#">options.coordRendering</a>	—	<p>Type: String</p> <p>String ID defining the algorithm for recalculating geometry coordinates as pixel coordinates. For the "Rectangle" geometry, can accept one of three values:</p> <ul style="list-style-type: none"> <li>shortestPath - Algorithm that considers projection cycling on the axes and generates pixel coordinates so that the distance between opposite corners is minimal.</li> <li>straightPath - Algorithm that does not consider projection cycling.</li> <li>boundsPath - Algorithm that interprets the coordinates of rectangle corners as coordinates corresponding to the lower and upper corners of a bounding area. When calculating diagonals for projections with cycled axes, the counter-clockwise direction is always used.</li> </ul>
<a href="#">options.geodesic</a>	false	<p>Type: Boolean</p> <p>Enables display using geodesic lines.</p>
<a href="#">options.pixelRendering</a>	"jumpy"	<p>Type: String</p> <p>Method for calculating pixel coordinates of the shape in cycled projections. This option accepts one of the following values:</p> <ul style="list-style-type: none"> <li>jumpy - The shape is placed as close as possible to the center of the map viewport, and can "jump" when the map is being moved.</li> <li>static - The shape is always located in the initial world and does not move when the map is moved.</li> </ul>
<a href="#">options.projection</a>	—	<p>Type: <a href="#">IProjection</a></p> <p>Projection.</p>

**Example:**

```
// Creating an instance of the point geometry (specifying coordinates).
var rectangleGeometry = new ymaps.geometry.Rectangle([[30, 50], [31, 51]]);
// Instantiating the geo object and passing our geometry to the constructor.
var rectangleGeoObject = new ymaps.GeoObject({ geometry: rectangleGeometry });

// Changing the coordinates via the geo object's "geometry" property.
rectangleGeoObject.geometry.setCoordinates([[10, 20], [51, 71]]);
// Or directly.
rectangleGeometry.setCoordinates([[10, 20], [51, 71]]);
// You can also access rectangleGeometry via rectangleGeoObject.geometry.
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

**Events**

Name	Description
<a href="#">change</a>	Change to corner coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>oldCoordinates</code> - Old coordinates of the corners.</li> <li><code>newCoordinates</code> - New coordinates of the corners.</li> </ul> Inherited from <a href="#">IRectangleGeometryAccess</a> .
<a href="#">mapchange</a>	Map reference changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>oldMap</code> - Old map.</li> <li><code>newMap</code> - New map.</li> </ul> Inherited from <a href="#">IGeometry</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">pixelgeometrychange</a>	The pixel geometry changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>pixelGeometry</code> - New <a href="#">IPixelGeometry</a> pixel geometry.</li> </ul> Inherited from <a href="#">IGeometry</a> .

**Methods**

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the rectangle.  Inherited from <a href="#">IRectangleGeometryAccess</a> .
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	Switches the object to "frozen" mode.  Inherited from <a href="#">IFreezable</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner.  Inherited from <a href="#">IGeometry</a> .

Name	Returns	Description
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for the point nearest to "anchorPosition" on the rectangle contour.  Inherited from <a href="#">IRectangleGeometryAccess</a> .
<a href="#">getCoordinates()</a>	Number[][]	Returns coordinates of two opposite corners of the rectangle.  Inherited from <a href="#">IRectangleGeometryAccess</a> .
<a href="#">getMap()</a>	Map null	Returns the current map.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getPixelGeometry([options])</a>	<a href="#">IPixelGeometry</a>	Returns the pixel geometry corresponding to the given geometry, its options, and the map state.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getType()</a>	String	Returns the "Rectangle" string.  Inherited from <a href="#">IRectangleGeometry</a> .
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false.  Inherited from <a href="#">IFreezable</a> .
<a href="#">setCoordinates(coordinates)</a>	<a href="#">IRectangleGeometryAccess</a>	Sets the coordinates of two opposite corners of the rectangle.  Inherited from <a href="#">IRectangleGeometryAccess</a> .
<a href="#">setMap(map)</a>		Sets the map.  Inherited from <a href="#">IGeometry</a> .
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode.  Inherited from <a href="#">IFreezable</a> .

## geometryEditor

### geometryEditor.Circle

Extends [IGeometryEditor](#).

The "Circle" geometry editor.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
geometryEditor.Circle(geometry[, options])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">ICircleGeometry</a> The "Circle" geometry.
<a href="#">options</a>	—	Type: Object Options for the geometry editor.
<a href="#">options.drawingCursor</a>	"arrow"	Type: Boolean The mouse cursor in drawing mode.
<a href="#">options.drawOver</a>	true	Type: Boolean Allows to put points on top of map objects in drawing mode.

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>	The geometry being edited. Inherited from <a href="#">IGeometryEditor</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">state</a>	<a href="#">IDataManager</a>	Manager for the state of the geometry editor. Data fields that are available via the "get" and "set" methods: <ul style="list-style-type: none"> <li>• <code>editing</code> - Checks whether editing mode is enabled. Type - Boolean. Default value - false.</li> <li>• <code>drawing</code> - Checks whether drawing mode is enabled. Type - Boolean. Default value - false.</li> </ul>

**Events**

Name	Description
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">statechange</a>	Change to the geometry editor state. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">IGeometryEditor</a> .

## Methods

Name	Returns	Description
<a href="#">startDrawing()</a>	<a href="#">vow.Promise</a>	Enables the mode for drawing a circle.
<a href="#">startEditing()</a>		Enables editing mode. Inherited from <a href="#">IGeometryEditor</a> .
<a href="#">stopDrawing()</a>	<a href="#">vow.Promise</a>	Disables the mode for drawing a circle.
<a href="#">stopEditing()</a>		Disables editing mode. Inherited from <a href="#">IGeometryEditor</a> .

## Fields details

### state

```
{IDataManager} state
```

Manager for the state of the geometry editor.

Data fields that are available via the "get" and "set" methods:

- `editing` - Checks whether editing mode is enabled. Type - Boolean. Default value - false.
- `drawing` - Checks whether drawing mode is enabled. Type - Boolean. Default value - false.

## Methods details

### startDrawing

```
{vow.Promise} startDrawing()
```

Enables the mode for drawing a circle.

**Returns** Promise object.

### stopDrawing

```
{vow.Promise} stopDrawing()
```

Disables the mode for drawing a circle.

**Returns** Promise object.

## geometryEditor.LineString

Extends [IGeometryEditor](#).

The "Polyline" geometry editor.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
geometryEditor.LineString(geometry[, options])
```

### Parameters:

Parameter	Default value	Description
<a href="#">geometry *</a>	—	Type: <a href="#">ILineStringGeometry</a>  "Polyline" geometry.
<a href="#">options</a>	—	Type: Object  Options for the geometry editor.
<a href="#">options.dblClickHandler</a>	—	Type: Function  Handler for double-clicking a vertex. Accepts a reference to a model of the vertex being edited. By default, the handler is defined by the function that removes the corresponding vertex.
<a href="#">options.drawingCursor</a>	"arrow"	Type: Boolean  The mouse cursor in the mode for adding new vertexes.
<a href="#">options.drawOver</a>	true	Type: Boolean  Allows to put points on top of map objects in the mode for adding new vertexes.
<a href="#">options.edgeInteractiveOptions</a>	true	Type: Boolean  Allows the intermediate vertices placemarks to use options with postfixes linked to the current state of the placemark. The following postfixes are available: <ul style="list-style-type: none"> <li>• Hover - Options with this given postfix are used when the user hovers over a placemark with the mouse pointer.</li> <li>• Drag - Options with this given postfix are used when the user drags a placemark.</li> </ul> Examples of such options: <code>edgeLayoutHover</code> , <code>edgeIconImageSizeActive</code> , <code>edgeIconImageShapeHover</code> , etc. If you do not want to change the options of intermediate vertices placemarks depending on their state, then you will need to disable this option.
<a href="#">options.edgeLayout</a>	—	Type: Function  Class of the layout for interim placemarks.
<a href="#">options.maxPoints</a>	Infinity	Type: Number  The maximum allowable number of vertexes on a polyline.

Parameter	Default value	Description
<a href="#">options.menuManager</a>	—	<p>Type: Function</p> <p>Context menu dispatcher for the menu that opens when clicking on a vertex. Accepts two arguments:</p> <ul style="list-style-type: none"> <li>• An array of objects describing the context menu items for this vertex.</li> <li>• A reference to a model of the vertex being edited.</li> </ul> <p>The dispatcher can change data in the passed array. Must return an array of objects that describe context menu items.</p>
<a href="#">options.minPoints</a>	0	<p>Type: Number</p> <p>The minimum allowable number of vertexes on a polyline.</p>
<a href="#">options.useAutoPanInDrawing</a>	true	<p>Type: Boolean</p> <p>Enables autopan of the map when dragging a vertex on the boundary.</p>
<a href="#">options.useMapMarginInDrawing</a>	true	<p>Type: Boolean</p> <p>Whether to use map margins in drawing mode.</p>
<a href="#">options.vertexInteractiveOptions</a>	true	<p>Type: Boolean</p> <p>Allows the vertices placemarks to use options with postfixes linked to the current state of the vertex. The following postfixes are available:</p> <ul style="list-style-type: none"> <li>• Hover - Options with this given postfix are used when the user hovers over a vertex with the mouse pointer.</li> <li>• Drag - Options with this given postfix are used when the user drags a vertex.</li> <li>• Active - Options with this given postfix are used when the context menu is opened for the vertex.</li> </ul> <p>Examples of such options: <code>vertexLayoutHover</code>, <code>vertexIconImageSizeActive</code>, <code>vertexIconImageShapeHover</code>, etc. If you do not want to change the options of vertices placemarks depending on their state, then you will need to disable this option.</p>
<a href="#">options.vertexLayout</a>	—	<p>Type: Function</p> <p>Class of the layout for placemarks on polyline vertexes.</p>

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>	The geometry being edited. Inherited from <a href="#">IGeometryEditor</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">state</a>	<a href="#">IDataManager</a>	Manager for the state of the geometry editor. Data fields that are available via the "get" and "set" methods: <ul style="list-style-type: none"> <li>• editing - Checks whether editing mode is enabled. Type - Boolean. Default value - false.</li> <li>• drawing - Checks whether vertex drawing mode is enabled. Type - Boolean. Default value - false.</li> <li>• drawingFrom - Checks how new points are added in drawing mode. Accepts one of two string values: "begin" - points are added at the beginning of the polyline; "end" - points are added at the end. Default value - "end".</li> </ul>

**Events**

Name	Description
<a href="#">beforeedgedrag</a>	Event preceding the "edgedrag" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>• pixelOffset - Array of two numbers that describe the pixel offset at this step.</li> <li>• edgeModel - Reference to the model of the draggable placemark.</li> <li>• globalPixels - Coordinates of the draggable placemark, in global pixel coordinates.</li> </ul> Names of methods that are accessible via <a href="#">Event.callMethod</a> : <ul style="list-style-type: none"> <li>• setPixelOffset - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.</li> </ul> If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent "edgedrag" event will be canceled.
<a href="#">beforeedgedragstart</a>	Event preceding the "edgedragstart" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>• domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> <li>• edgeModel - Reference to the model of the draggable placemark.</li> <li>• globalPixels - Coordinates of the draggable placemark, in global pixel coordinates.</li> </ul> If the <a href="#">Event.preventDefault</a> method is called for this event, any subsequent dragging, as well as the "edgedragstart" event, will be canceled.

Name	Description
<a href="#">beforevertexadd</a>	<p>Event preceding the "vertexadd" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>parentModel - Reference to the parent data model for the vertex being added.</li> <li>vertexIndex - Index of the vertex being added.</li> <li>globalPixels - Coordinates of the vertex being added, in global pixel coordinates.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>setGlobalPixels - Use this method to correct the coordinate values of the added vertex. It takes an argument with the new global pixel coordinates of the vertex as an array of two numbers.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent "vertexadd" event will be canceled.</p>
<a href="#">beforevertexdrag</a>	<p>Event preceding the "vertexdrag" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>pixelOffset - Array of two numbers that describe the pixel offset at this step.</li> <li>vertexModel - Reference to the model of the draggable vertex.</li> <li>globalPixels - Coordinates of the draggable vertex, in global pixel coordinates.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>setPixelOffset - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent "vertexdrag" event will be canceled.</p>
<a href="#">beforevertexdragstart</a>	<p>Event preceding the "vertexdragstart" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> <li>vertexModel - Reference to the model of the draggable vertex.</li> <li>globalPixels - Coordinates of the draggable vertex, in global pixel coordinates.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, any subsequent dragging, as well as the "vertexdragstart" event, will be canceled.</p>
<a href="#">beforevertexdraw</a>	<p>Event preceding the "vertexdraw" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>parentModel - Reference to the parent data model for the vertex being added.</li> <li>vertexIndex - Index of the vertex being added.</li> <li>globalPixels - Coordinates of the vertex being added, in global pixel coordinates.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>setGlobalPixels - Use this method to correct the coordinate values of the added vertex. It takes an argument with the new global pixel coordinates of the vertex as an array of two numbers.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent "vertexdraw" event will be canceled.</p>
<a href="#">drawingstart</a>	Enabling the mode for adding new vertexes. Instance of the <a href="#">Event</a> class.
<a href="#">drawingstop</a>	Disabling the mode for adding new vertexes. Instance of the <a href="#">Event</a> class.

Name	Description
<a href="#">edgedrag</a>	<p>Dragging an interim placemark. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>pixelOffset - Array of two numbers that describe the pixel offset at this step.</li> <li>edgeModel - Reference to the model of the draggable placemark.</li> <li>globalPixels - Coordinates of the draggable placemark, in global pixel coordinates.</li> </ul>
<a href="#">edgedragend</a>	<p>End of dragging an interim placemark. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>edgeModel - Reference to the model of the draggable placemark.</li> <li>globalPixels - Coordinates of the draggable placemark, in global pixel coordinates.</li> </ul>
<a href="#">edgedragstart</a>	<p>Start of dragging an interim placemark. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> <li>edgeModel - Reference to the model of the draggable placemark.</li> <li>globalPixels - Coordinates of the draggable placemark, in global pixel coordinates.</li> </ul>
<a href="#">editingstart</a>	Enabling the mode for editing vertexes. Instance of the <a href="#">Event</a> class.
<a href="#">editingstop</a>	Disabling the mode for editing vertexes. Instance of the <a href="#">Event</a> class.
<a href="#">framingstart</a>	Enabling the zoom mode. Instance of the <a href="#">Event</a> class.
<a href="#">framingstop</a>	Disabling the zoom mode. Instance of the <a href="#">Event</a> class.
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">statechange</a>	<p>Change to the geometry editor state. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IGeometryEditor</a>.</p>
<a href="#">vertexadd</a>	<p>Adding a new vertex. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>parentModel - Reference to the parent data model for the vertex that was added.</li> <li>vertexIndex - Index of the vertex that was added.</li> <li>globalPixels - Coordinates of the vertex that was added, in global pixel coordinates.</li> </ul>
<a href="#">vertexdrag</a>	<p>Dragging a vertex. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>pixelOffset - Array of two numbers that describe the pixel offset at this step.</li> <li>vertexModel - Reference to the model of the draggable vertex.</li> <li>globalPixels - Coordinates of the draggable vertex, in global pixel coordinates.</li> </ul>
<a href="#">vertexdragend</a>	<p>End of vertex dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>vertexModel - Reference to the model of the draggable vertex.</li> <li>globalPixels - Coordinates of the draggable vertex, in global pixel coordinates.</li> </ul>

Name	Description
<a href="#">vertexdragstart</a>	<p>Start of vertex dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> <li>• <code>vertexModel</code> - Reference to the model of the draggable vertex.</li> <li>• <code>globalPixels</code> - Coordinates of the draggable vertex, in global pixel coordinates.</li> </ul>
<a href="#">vertexdraw</a>	<p>Drawing a new vertex. This event usually precedes the add vertex event, and occurs when the mouse moves and the mode for adding new vertexes is enabled. Based on data passed in this event, guide lines are displayed in vertex drawing mode. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>parentModel</code> - Reference to the parent data model for the vertex being added.</li> <li>• <code>vertexIndex</code> - Index of the vertex being added.</li> <li>• <code>globalPixels</code> - Coordinates of the vertex being added, in global pixel coordinates.</li> </ul>

## Methods

Name	Returns	Description
<a href="#">getModel()</a>	<a href="#">vow.Promise</a>	<p>Returns the promise object, which is confirmed by the model object at the time it is actually created, or is rejected with one of the following error messages:</p> <ul style="list-style-type: none"> <li>• Canceled - Editing mode is disabled until the model is actually created.</li> <li>• Editor wasn't started - Editing mode is not enabled.</li> </ul>
<a href="#">getModelSync()</a>	<a href="#">geometryEditor.model.RootLineSegment</a>   null	<p>Returns the editor's data model, or null if it is missing at the time of the call.</p>
<a href="#">getView()</a>	<a href="#">vow.Promise</a>	<p>Returns the promise object, which is confirmed by the representation object at the time it is actually created, or is rejected with one of the following error messages:</p> <ul style="list-style-type: none"> <li>• Canceled - Editing mode is disabled until the representation is actually created.</li> <li>• Editor wasn't started - Editing mode is not enabled.</li> </ul>
<a href="#">getViewSync()</a>	<a href="#">geometryEditor.view.Path</a>   null	<p>Returns the editor's representation, or null if it is missing at the time of the call.</p>

Name	Returns	Description
<a href="#">startDrawing()</a>	<a href="#">vow.Promise</a>	Enables vertex drawing mode for a polyline. Enabling occurs asynchronously.
<a href="#">startEditing()</a>	<a href="#">vow.Promise</a>	Enables editing mode. Enabling occurs asynchronously.
<a href="#">startFraming()</a>	<a href="#">vow.Promise</a>	Enables the zoom mode for a polyline. Enabling occurs asynchronously.
<a href="#">stopDrawing()</a>		Disables vertex drawing mode for a polyline.
<a href="#">stopEditing()</a>		Disables editing mode.
<a href="#">stopFraming()</a>		Disables the zoom mode.

### Fields details

#### state

```
{IDataManager} state
```

Manager for the state of the geometry editor.

Data fields that are available via the "get" and "set" methods:

- `editing` - Checks whether editing mode is enabled. Type - Boolean. Default value - `false`.
- `drawing` - Checks whether vertex drawing mode is enabled. Type - Boolean. Default value - `false`.
- `drawingFrom` - Checks how new points are added in drawing mode. Accepts one of two string values: "begin" - points are added at the beginning of the polyline; "end" - points are added at the end. Default value - "end".

### Events details

#### beforeedgedrag

Event preceding the "edgedrag" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `pixelOffset` - Array of two numbers that describe the pixel offset at this step.
- `edgeModel` - Reference to the model of the draggable placemark.
- `globalPixels` - Coordinates of the draggable placemark, in global pixel coordinates.

Names of methods that are accessible via [Event.callMethod](#):

- `setPixelOffset` - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the [Event.preventDefault](#) method is called for this event, a subsequent "edgedrag" event will be canceled.

#### beforeedgedragstart

Event preceding the "edgedragstart" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.
- `edgeModel` - Reference to the model of the draggable placemark.
- `globalPixels` - Coordinates of the draggable placemark, in global pixel coordinates.

If the [Event.preventDefault](#) method is called for this event, any subsequent dragging, as well as the "edgedragstart" event, will be canceled.

### beforevertexadd

Event preceding the "vertexadd" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `parentModel` - Reference to the parent data model for the vertex being added.
- `vertexIndex` - Index of the vertex being added.
- `globalPixels` - Coordinates of the vertex being added, in global pixel coordinates.

Names of methods that are accessible via [Event.callMethod](#):

- `setGlobalPixels` - Use this method to correct the coordinate values of the added vertex. It takes an argument with the new global pixel coordinates of the vertex as an array of two numbers.

If the [Event.preventDefault](#) method is called for this event, a subsequent "vertexadd" event will be canceled.

### Example:

```
// Correcting the coordinates of the vertex add event so that they stay within a square with sides that are
// 100 pixels that is centered on the map center.
polyline.editor.events.add(["beforevertexdraw", "beforevertexadd"], function (event) {
  var mapGlobalPixelCenter = geoMap.getGlobalPixelCenter();
  var globalPixels = event.get("globalPixels");
  var pixelBounds = [
    [mapGlobalPixelCenter[0] - 100, mapGlobalPixelCenter[1] - 100],
    [mapGlobalPixelCenter[0] + 100, mapGlobalPixelCenter[1] + 100]
  ];
  event.callMethod("setGlobalPixels", [
    Math.max(Math.min(globalPixels[0], pixelBounds[1][0]), pixelBounds[0][0]),
    Math.max(Math.min(globalPixels[1], pixelBounds[1][1]), pixelBounds[0][1])
  ]);
});
```

### beforevertexdrag

Event preceding the "vertexdrag" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `pixelOffset` - Array of two numbers that describe the pixel offset at this step.
- `vertexModel` - Reference to the model of the draggable vertex.
- `globalPixels` - Coordinates of the draggable vertex, in global pixel coordinates.

Names of methods that are accessible via [Event.callMethod](#):

- `setPixelOffset` - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the [Event.preventDefault](#) method is called for this event, a subsequent "vertexdrag" event will be canceled.

### Example:

```
// Inverting the offset when dragging the vertex.
polyline.editor.events.add("beforevertexdrag", function (event) {
  var pixelOffset = event.get("pixelOffset");
  event.callMethod("setPixelOffset", [-pixelOffset[0], -pixelOffset[1]]);
});
```

### beforevertexdragstart

Event preceding the "vertexdragstart" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.
- `vertexModel` - Reference to the model of the draggable vertex.
- `globalPixels` - Coordinates of the draggable vertex, in global pixel coordinates.

If the [Event.preventDefault](#) method is called for this event, any subsequent dragging, as well as the "vertexdragstart" event, will be canceled.

**beforevertexdraw**

Event preceding the "vertexdraw" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- parentModel - Reference to the parent data model for the vertex being added.
- vertexIndex - Index of the vertex being added.
- globalPixels - Coordinates of the vertex being added, in global pixel coordinates.

Names of methods that are accessible via [Event.callMethod](#):

- setGlobalPixels - Use this method to correct the coordinate values of the added vertex. It takes an argument with the new global pixel coordinates of the vertex as an array of two numbers.

If the [Event.preventDefault](#) method is called for this event, a subsequent "vertexdraw" event will be canceled.

**drawingstart**

Enabling the mode for adding new vertexes. Instance of the [Event](#) class.

**drawingstop**

Disabling the mode for adding new vertexes. Instance of the [Event](#) class.

**edgedrag**

Dragging an interim placemark. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- pixelOffset - Array of two numbers that describe the pixel offset at this step.
- edgeModel - Reference to the model of the draggable placemark.
- globalPixels - Coordinates of the draggable placemark, in global pixel coordinates.

**edgedragend**

End of dragging an interim placemark. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- edgeModel - Reference to the model of the draggable placemark.
- globalPixels - Coordinates of the draggable placemark, in global pixel coordinates.

**edgedragstart**

Start of dragging an interim placemark. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- domEvent - Source DOM event (as a [DomEvent](#) object), if there is one.
- edgeModel - Reference to the model of the draggable placemark.
- globalPixels - Coordinates of the draggable placemark, in global pixel coordinates.

**editingstart**

Enabling the mode for editing vertexes. Instance of the [Event](#) class.

**editingstop**

Disabling the mode for editing vertexes. Instance of the [Event](#) class.

**framingstart**

Enabling the zoom mode. Instance of the [Event](#) class.

### framingstop

Disabling the zoom mode. Instance of the [Event](#) class.

### vertexadd

Adding a new vertex. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `parentModel` - Reference to the parent data model for the vertex that was added.
- `vertexIndex` - Index of the vertex that was added.
- `globalPixels` - Coordinates of the vertex that was added, in global pixel coordinates.

### vertexdrag

Dragging a vertex. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `pixelOffset` - Array of two numbers that describe the pixel offset at this step.
- `vertexModel` - Reference to the model of the draggable vertex.
- `globalPixels` - Coordinates of the draggable vertex, in global pixel coordinates.

### vertexdragend

End of vertex dragging. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `vertexModel` - Reference to the model of the draggable vertex.
- `globalPixels` - Coordinates of the draggable vertex, in global pixel coordinates.

### vertexdragstart

Start of vertex dragging. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.
- `vertexModel` - Reference to the model of the draggable vertex.
- `globalPixels` - Coordinates of the draggable vertex, in global pixel coordinates.

### vertexdraw

Drawing a new vertex. This event usually precedes the add vertex event, and occurs when the mouse moves and the mode for adding new vertexes is enabled. Based on data passed in this event, guide lines are displayed in vertex drawing mode. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `parentModel` - Reference to the parent data model for the vertex being added.
- `vertexIndex` - Index of the vertex being added.
- `globalPixels` - Coordinates of the vertex being added, in global pixel coordinates.

## Methods details

### getModel

```
{vow.Promise} getModel()
```

**Returns** the promise object, which is confirmed by the model object at the time it is actually created, or is rejected with one of the following error messages:

- `Canceled` - Editing mode is disabled until the model is actually created.
- `Editor wasn't started` - Editing mode is not enabled.

### getModelSync

```
{geometryEditor.model.RootLineString|null} getModelSync()
```

**Returns** the editor's data model, or null if it is missing at the time of the call.

### getView

```
{vow.Promise} getView()
```

**Returns** the promise object, which is confirmed by the representation object at the time it is actually created, or is rejected with one of the following error messages:

- Canceled - Editing mode is disabled until the representation is actually created.
- Editor wasn't started - Editing mode is not enabled.

### getViewSync

```
{geometryEditor.view.Path|null} getViewSync()
```

**Returns** the editor's representation, or null if it is missing at the time of the call.

### startDrawing

```
{vow.Promise} startDrawing()
```

Enables vertex drawing mode for a polyline. Enabling occurs asynchronously.

**Returns** the promise object, which is confirmed when drawing mode has actually started, or is rejected with one of the following error messages:

- Canceled - Drawing mode is disabled until it is actually started.

### startEditing

```
{vow.Promise} startEditing()
```

Enables editing mode. Enabling occurs asynchronously.

**Returns** the promise object, which is confirmed when editing mode has actually started, or is rejected with one of the following error messages:

- Canceled - Editing mode is disabled until it is actually started.

### startFraming

```
{vow.Promise} startFraming()
```

Enables the zoom mode for a polyline. Enabling occurs asynchronously.

**Returns** the promise object that is confirmed when zoom mode actually starts.

### stopDrawing

```
{ } stopDrawing()
```

Disables vertex drawing mode for a polyline.

### stopEditing

```
{ } stopEditing()
```

Disables editing mode.

### stopFraming

```
{ } stopFraming()
```

Disables the zoom mode.

## geometryEditor.model

### geometryEditor.model.ChildLinearRing

**Note:** The constructor of the `geometryEditor.model.ChildLinearRing` class is hidden, as this class is not intended for autonomous initialization.

Extends [geometryEditor.model.ChildLineString](#).

Model for a child closed contour. The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand.

[Fields](#) | [Methods](#)

### Fields

Name	Type	Description
<a href="#">editor</a>	<a href="#">IGeometryEditor</a>	Geometry editor. Inherited from <a href="#">IGeometryEditorChildModel</a> .
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IBaseGeometry</a>	Geometry of the model. Inherited from <a href="#">IGeometryEditorChildModel</a> .

### Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destructor. Inherited from <a href="#">IGeometryEditorModel</a> .
<a href="#">getAllVerticesNumber()</a>	Integer	Returns the total number of vertexes in the geometry being edited. Inherited from <a href="#">geometryEditor.model.ChildLineString</a> .
<a href="#">getEdgeModels()</a>	<a href="#">geometryEditor.model.Edge[]</a>	Returns an array of models for interim placemarks. Inherited from <a href="#">geometryEditor.model.ChildLineString</a> .
<a href="#">getIndex()</a>	Integer	Returns the index of the child polyline in the parent model. Inherited from <a href="#">geometryEditor.model.ChildLineString</a> .
<a href="#">getParent()</a>	<a href="#">IGeometryEditorModel</a>	Returns the parent data model. Inherited from <a href="#">IGeometryEditorChildModel</a> .

Name	Returns	Description
<a href="#">getPixels()</a>	Number[]	Returns the model's pixel data.  Inherited from <a href="#">IGeometryEditorModel</a> .
<a href="#">getVertexModels()</a>	<a href="#">geometryEditor.model.ChildVertex</a> []	Returns an array of models for child vertexes.  Inherited from <a href="#">geometryEditor.model.ChildLineString</a> .
<a href="#">setIndex(index)</a>		Sets the index of the child polyline in the parent model.  Inherited from <a href="#">geometryEditor.model.ChildLineString</a> .
<a href="#">setPixels(pixels)</a>		Sets the model's pixel data.  Inherited from <a href="#">IGeometryEditorChildModel</a> .
<a href="#">spliceVertices(start, deleteCount)</a>	Number[][]	Deletes a defined number of polyline vertexes, starting from the specified index. New vertexes can be added in place of the deleted ones. Global pixel coordinates of the new vertexes can be passed as additional arguments after the "deleteCount" parameter.  Inherited from <a href="#">geometryEditor.model.ChildLineString</a> .

### **geometryEditor.model.ChildLineString**

**Note:** The constructor of the `geometryEditor.model.ChildLineString` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeometryEditorChildModel](#).

Model of the child polyline. The constructor is not available in the `package.full` (a standard set of modules). This module is loaded on demand.

[Fields](#) | [Methods](#)

#### **Fields**

Name	Type	Description
<a href="#">editor</a>	<a href="#">IGeometryEditor</a>	Geometry editor.  Inherited from <a href="#">IGeometryEditorChildModel</a> .
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IBaseGeometry</a>	Geometry of the model.  Inherited from <a href="#">IGeometryEditorChildModel</a> .

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destructor. Inherited from <a href="#">IGeometryEditorModel</a> .
<a href="#">getAllVerticesNumber()</a>	Integer	Returns the total number of vertexes in the geometry being edited.
<a href="#">getEdgeModels()</a>	<a href="#">geometryEditor.model.Edge[]</a>	Returns an array of models for interim placemarks.
<a href="#">getIndex()</a>	Integer	Returns the index of the child polyline in the parent model.
<a href="#">getParent()</a>	<a href="#">IGeometryEditorModel</a>	Returns the parent data model. Inherited from <a href="#">IGeometryEditorChildModel</a> .
<a href="#">getPixels()</a>	Number[]	Returns the model's pixel data. Inherited from <a href="#">IGeometryEditorModel</a> .
<a href="#">getVertexModels()</a>	<a href="#">geometryEditor.model.ChildVertex[]</a>	Returns an array of models for child vertexes.
<a href="#">setIndex(index)</a>		Sets the index of the child polyline in the parent model.
<a href="#">setPixels(pixels)</a>		Sets the model's pixel data. Inherited from <a href="#">IGeometryEditorChildModel</a> .
<a href="#">spliceVertices(start, deleteCount)</a>	Number[][]	Deletes a defined number of polyline vertexes, starting from the specified index. New vertexes can be added in place of the deleted ones. Global pixel coordinates of the new vertexes can be passed as additional arguments after the "deleteCount" parameter.

## Methods details

**getAllVerticesNumber**

```
{Integer} getAllVerticesNumber()
```

**Returns** the total number of vertexes in the geometry being edited.

**getEdgeModels**

```
{geometryEditor.model.Edge[]} getEdgeModels()
```

**Returns** an array of models for interim placemarks.

**getIndex**

```
{Integer} getIndex()
```

**Returns** the index of the child polyline in the parent model.

**getVertexModels**

```
{geometryEditor.model.ChildVertex[]} getVertexModels()
```

**Returns** an array of models for child vertexes.

**setIndex**

```
{ } setIndex(index)
```

Sets the index of the child polyline in the parent model.

**Parameters:**

Parameter	Default value	Description
<code>index</code> *	—	Type: Integer Index of a child vertex.

\* Mandatory parameter/option.

**spliceVertices**

```
{Number[][]} spliceVertices(start, deleteCount)
```

Deletes a defined number of polyline vertexes, starting from the specified index. New vertexes can be added in place of the deleted ones. Global pixel coordinates of the new vertexes can be passed as additional arguments after the "deleteCount" parameter.

**Returns** an array of coordinates of deleted vertexes.

**Parameters:**

Parameter	Default value	Description
<code>start</code> *	—	Type: Integer The index to start from for removing and adding vertexes.
<code>deleteCount</code> *	—	Type: Integer The number of deleted vertexes.

\* Mandatory parameter/option.

**geometryEditor.model.ChildVertex**

**Note:** The constructor of the `geometryEditor.model.ChildVertex` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeometryEditorChildModel](#).

Model for a child index. The constructor is not available in the `package.full` (a standard set of modules). This module is loaded on demand.

[Fields](#) | [Methods](#)**Fields**

Name	Type	Description
<a href="#">editor</a>	<a href="#">IGeometryEditor</a>	Geometry editor. Inherited from <a href="#">IGeometryEditorChildModel</a> .
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IBaseGeometry</a>	Geometry of the model. Inherited from <a href="#">IGeometryEditorChildModel</a> .

**Methods**

Name	Returns	Description
<a href="#">destroy()</a>		Destructor. Inherited from <a href="#">IGeometryEditorModel</a> .
<a href="#">getAllVerticesNumber()</a>	Integer	Returns the total number of vertexes in the geometry being edited.
<a href="#">getIndex()</a>	Integer	Returns the index of the child vertex in the parent model.
<a href="#">getNextVertex()</a>	<a href="#">geometryEditor.model.ChildVertex</a> null	Returns a reference to the model for the next vertex.
<a href="#">getParent()</a>	<a href="#">IGeometryEditorModel</a>	Returns the parent data model. Inherited from <a href="#">IGeometryEditorChildModel</a> .
<a href="#">getPixels()</a>	Number[]	Returns the model's pixel data. Inherited from <a href="#">IGeometryEditorModel</a> .
<a href="#">getPrevVertex()</a>	<a href="#">geometryEditor.model.ChildVertex</a> null	Returns a reference to the model for the previous vertex.
<a href="#">setGlobalPixels(pixels)</a>		Sets global pixel coordinates of the vertex.
<a href="#">setIndex(index)</a>		Sets the index of the child vertex in the parent model.
<a href="#">setNextVertex(nextVertex)</a>		Sets the reference to the model for the next vertex.
<a href="#">setPixels(pixels)</a>		Sets the model's pixel data. Inherited from <a href="#">IGeometryEditorChildModel</a> .

Name	Returns	Description
<code>setPrevVertex(prevVertex)</code>		Sets the reference to the model for the previous vertex.

## Methods details

### getAllVerticesNumber

```
{Integer} getAllVerticesNumber()
```

**Returns** the total number of vertexes in the geometry being edited.

### getIndex

```
{Integer} getIndex()
```

**Returns** the index of the child vertex in the parent model.

### getNextVertex

```
{geometryEditor.model.ChildVertex|null} getNextVertex()
```

**Returns** a reference to the model for the next vertex.

### getPrevVertex

```
{geometryEditor.model.ChildVertex|null} getPrevVertex()
```

**Returns** a reference to the model for the previous vertex.

### setGlobalPixels

```
{ } setGlobalPixels(pixels)
```

Sets global pixel coordinates of the vertex.

#### Parameters:

Parameter	Default value	Description
<code>pixels *</code>	—	Type: Number[]  Global pixel coordinates of the vertex.

\* Mandatory parameter/option.

### setIndex

```
{ } setIndex(index)
```

Sets the index of the child vertex in the parent model.

#### Parameters:

Parameter	Default value	Description
<code>index *</code>	—	Type: Integer  Index of a child vertex.

\* Mandatory parameter/option.

### setNextVertex

```
{ } setNextVertex (nextVertex)
```

Sets the reference to the model for the next vertex.

#### Parameters:

Parameter	Default value	Description
<a href="#">nextVertex</a> *	—	Type: <a href="#">geometryEditor.model.ChildVertex</a>   null  Model for the next vertex.

\* Mandatory parameter/option.

### setPrevVertex

```
{ } setPrevVertex (prevVertex)
```

Sets the reference to the model for the previous vertex.

#### Parameters:

Parameter	Default value	Description
<a href="#">prevVertex</a> *	—	Type: <a href="#">geometryEditor.model.ChildVertex</a>   null  Model for the previous vertex.

\* Mandatory parameter/option.

### geometryEditor.model.Edge

**Note:** The constructor of the `geometryEditor.model.Edge` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeometryEditorRootModel](#).

Interim placemark model. The constructor is not available in the `package.full` (a standard set of modules). This module is loaded on demand.

[Fields](#) | [Methods](#)

#### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destructor. Inherited from <a href="#">IGeometryEditorModel</a> .
<a href="#">getNextVertex()</a>	<a href="#">geometryEditor.model.ChildVertex</a> or null	Returns a reference to the model for the next vertex.
<a href="#">getPixels()</a>	Number[]	Returns the model's pixel data. Inherited from <a href="#">IGeometryEditorModel</a> .
<a href="#">getPrevVertex()</a>	<a href="#">geometryEditor.model.ChildVertex</a> or null	Returns a reference to the model for the previous vertex.
<a href="#">setNextVertex(nextVertex)</a>		Sets the reference to the model for the next vertex.
<a href="#">setPrevVertex(prevVertex)</a>		Sets the reference to the model for the previous vertex.

## Methods details

**getNextVertex**

```
{geometryEditor.model.ChildVertex|null} getNextVertex()
```

Returns a reference to the model for the next vertex.

**getPrevVertex**

```
{geometryEditor.model.ChildVertex|null} getPrevVertex()
```

Returns a reference to the model for the previous vertex.

**setNextVertex**

```
{ } setNextVertex(nextVertex)
```

Sets the reference to the model for the next vertex.

**Parameters:**

Parameter	Default value	Description
<a href="#">nextVertex</a> *	—	Type: <a href="#">geometryEditor.model.ChildVertex</a> or null  Model for the next vertex.

\* Mandatory parameter/option.

**setPrevVertex**

```
{ } setPrevVertex(prevVertex)
```

Sets the reference to the model for the previous vertex.

**Parameters:**

Parameter	Default value	Description
<a href="#">prevVertex</a> *	—	Type: <a href="#">geometryEditor.model.ChildVertex</a>   null  Model for the previous vertex.

\* Mandatory parameter/option.

**geometryEditor.model.EdgeGeometry**

**Note:** The constructor of the `geometryEditor.model.EdgeGeometry` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeometry](#).

Interim placemark geometry. The constructor is not available in the `package.full` (a standard set of modules). This module is loaded on demand.

[Fields](#) | [Events](#) | [Methods](#)

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

**Events**

Name	Description
<a href="#">mapchange</a>	Map reference changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>oldMap</code> - Old map.</li> <li><code>newMap</code> - New map.</li> </ul> Inherited from <a href="#">IGeometry</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">pixelgeometrychange</a>	The pixel geometry changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>pixelGeometry</code> - New <a href="#">IPixelGeometry</a> pixel geometry.</li> </ul> Inherited from <a href="#">IGeometry</a> .

**Methods**

Name	Returns	Description
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getMap()</a>	<a href="#">Map</a>  null	Returns the current map.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getPixelGeometry([options])</a>	<a href="#">IPixelGeometry</a>	Returns the pixel geometry corresponding to the given geometry, its options, and the map state.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getType()</a>	String	Returns ID of the geometry type.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">setMap(map)</a>		Sets the map.  Inherited from <a href="#">IGeometry</a> .

**geometryEditor.model.RootLineString**

**Note:** The constructor of the `geometryEditor.model.RootLineString` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeometryEditorRootModel](#).

Model of the root polyline. The constructor is not available in the `package.full` (a standard set of modules). This module is loaded on demand.

[Fields](#) | [Methods](#)

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

**Methods**

Name	Returns	Description
<a href="#">destroy()</a>		Destructor.  Inherited from <a href="#">IGeometryEditorModel</a> .
<a href="#">getAllVerticesNumber()</a>	Integer	Returns the total number of vertexes in the geometry being edited.

Name	Returns	Description
<a href="#">getPixels()</a>	Number[]	Returns the model's pixel data.  Inherited from <a href="#">IGeometryEditorModel</a> .
<a href="#">getVertexModels()</a>	<a href="#">geometryEditor.model.ChildVertex</a> []	Returns an array of models for child vertexes.
<a href="#">spliceVertices(start, deleteCount)</a>	Number[][]	Deletes a defined number of polyline vertexes, starting from the specified index. New vertexes can be added in place of the deleted ones. Global pixel coordinates of the new vertexes can be passed as additional arguments after the "deleteCount" parameter.

## Methods details

### getAllVerticesNumber

```
{Integer} getAllVerticesNumber()
```

**Returns** the total number of vertexes in the geometry being edited.

### getVertexModels

```
{geometryEditor.model.ChildVertex[]} getVertexModels()
```

**Returns** an array of models for child vertexes.

### spliceVertices

```
{Number[][]} spliceVertices(start, deleteCount)
```

Deletes a defined number of polyline vertexes, starting from the specified index. New vertexes can be added in place of the deleted ones. Global pixel coordinates of the new vertexes can be passed as additional arguments after the "deleteCount" parameter.

**Returns** an array of coordinates of deleted vertexes.

#### Parameters:

Parameter	Default value	Description
<a href="#">start</a> *	—	Type: Integer  The index to start from for removing and adding vertexes.
<a href="#">deleteCount</a> *	—	Type: Integer  The number of deleted vertexes.

\* Mandatory parameter/option.

**geometryEditor.model.RootPolygon**

**Note:** The constructor of the `geometryEditor.model.RootPolygon` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeometryEditorRootModel](#).

Model for the root polygon. The constructor is not available in the `package.full` (a standard set of modules). This module is loaded on demand.

[Fields](#) | [Methods](#)

**Fields**

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

**Methods**

Name	Returns	Description
<code>destroy()</code>		Destructor. Inherited from <a href="#">IGeometryEditorModel</a> .
<code>getAllVerticesNumber()</code>	Integer	Returns the total number of vertexes in the geometry being edited.
<code>getPathModels()</code>	<a href="#">geometryEditor.model.ChildLinearRing</a> []	Returns an array of models for child contours.
<code>getPixels()</code>	Number []	Returns the model's pixel data. Inherited from <a href="#">IGeometryEditorModel</a> .
<code>splicePaths(start, deleteCount)</code>	Number [] []	Deletes a defined number of polygon contours, starting from the specified index. New contours can be added in place of the deleted ones. Global pixel coordinates of the new contours can be passed as additional arguments after the "deleteCount" parameter.

**Methods details****getAllVerticesNumber**

```
{Integer} getAllVerticesNumber()
```

**Returns** the total number of vertexes in the geometry being edited.

**getPathModels**

```
{geometryEditor.model.ChildLinearRing[]} getPathModels()
```

**Returns** an array of models for child contours.

### splicePaths

```
{Number[][]} splicePaths(start, deleteCount)
```

Deletes a defined number of polygon contours, starting from the specified index. New contours can be added in place of the deleted ones. Global pixel coordinates of the new contours can be passed as additional arguments after the "deleteCount" parameter.

**Returns** an array of coordinates of deleted contours.

#### Parameters:

Parameter	Default value	Description
<a href="#">start</a> *	—	Type: Integer  The index to start from for removing and adding contours.
<a href="#">deleteCount</a> *	—	Type: Integer  The number of contours to be deleted.

\* Mandatory parameter/option.

## geometryEditor.Point

Extends [IGeometryEditor](#).

The "Point" geometry editor.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
geometryEditor.Point(geometry[, options])
```

#### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IPointGeometry</a>  The "Point" geometry.
<a href="#">options</a>	—	Type: Object  Options for the geometry editor.
<a href="#">options.dblClickHandler</a>	—	Type: Function  Handler for double-clicking a vertex. Accepts a reference to a model of the vertex being edited. By default, the handler is defined by the function that removes the corresponding vertex.
<a href="#">options.drawingCursor</a>	"arrow"	Type: Boolean  The mouse cursor in drawing mode.

Parameter	Default value	Description
<a href="#">options.drawOver</a>	true	Type: Boolean  Allows to put points on top of map objects in drawing mode.

\* Mandatory parameter/option.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>	The geometry being edited. Inherited from <a href="#">IGeometryEditor</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">state</a>	<a href="#">IDataManager</a>	Manager for the state of the geometry editor. Data fields that are available via the "get" and "set" methods: <ul style="list-style-type: none"> <li>• editing - Checks whether editing mode is enabled. Type - Boolean. Default value - false.</li> <li>• drawing - Checks whether drawing mode is enabled. Type - Boolean. Default value - false.</li> </ul>

### Events

Name	Description
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">statechange</a>	Change to the geometry editor state. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">IGeometryEditor</a> .

### Methods

Name	Returns	Description
<a href="#">startDrawing()</a>	<a href="#">vow.Promise</a>	Enables the mode for drawing points.
<a href="#">startEditing()</a>		Enables editing mode. Inherited from <a href="#">IGeometryEditor</a> .
<a href="#">stopDrawing()</a>	<a href="#">vow.Promise</a>	Disables the mode for drawing points.
<a href="#">stopEditing()</a>		Disables editing mode. Inherited from <a href="#">IGeometryEditor</a> .

## Fields details

### state

```
{IDataManager} state
```

Manager for the state of the geometry editor.

Data fields that are available via the "get" and "set" methods:

- editing - Checks whether editing mode is enabled. Type - Boolean. Default value - false.
- drawing - Checks whether drawing mode is enabled. Type - Boolean. Default value - false.

## Methods details

### startDrawing

```
{vow.Promise} startDrawing()
```

Enables the mode for drawing points.

**Returns** Promise object.

### stopDrawing

```
{vow.Promise} stopDrawing()
```

Disables the mode for drawing points.

**Returns** Promise object.

## geometryEditor.Polygon

Extends [IGeometryEditor](#).

The "Polygon" geometry editor.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
geometryEditor.Polygon(geometry[, options])
```

### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IPolygonGeometry</a> The "Polygon" geometry.
<a href="#">options</a>	—	Type: Object Options for the geometry editor.
<a href="#">options.dblClickHandler</a>	—	Type: Function Handler for double-clicking a vertex. Accepts a reference to a model of the vertex being edited. By default, the handler is defined by the function that removes the corresponding vertex.

Parameter	Default value	Description
<a href="#">options.drawingCursor</a>	"arrow"	Type: Boolean  The mouse cursor in the mode for adding new vertexes.
<a href="#">options.drawOver</a>	true	Type: Boolean  Allows to put points on top of map objects in the mode for adding new vertexes.
<a href="#">options.edgeInteractiveOptions</a>	true	Type: Boolean  Allows the intermediate vertices placemarks to use options with postfixes linked to the current state of the placemark. The following postfixes are available: <ul style="list-style-type: none"> <li>• Drag - Options with this given postfix are used when the user drags a placemark.</li> <li>• Hover - Options with this given postfix are used when the user hovers over a placemark with the mouse pointer.</li> </ul> Examples of such options: <code>edgeLayoutHover</code> , <code>edgeIconImageSizeActive</code> , <code>edgeIconImageShapeHover</code> , etc. If you do not want to change the options of intermediate vertices placemarks depending on their state, then you will need to disable this option.
<a href="#">options.edgeLayout</a>	—	Type: Function  Class of the layout for interim placemarks.
<a href="#">options.maxPoints</a>	Infinity	Type: Number  The maximum allowable number of vertexes on a polygon.
<a href="#">options.menuManager</a>	—	Type: Function  Context menu dispatcher for the menu that opens when clicking on a vertex. Accepts two arguments: <ul style="list-style-type: none"> <li>• An array of objects describing the context menu items for this vertex.</li> <li>• A reference to a model of the vertex being edited.</li> </ul> The dispatcher can change data in the passed array. Must return an array of objects that describe context menu items.

Parameter	Default value	Description
<a href="#">options.minPoints</a>	0	Type: Number  The minimum allowable number of vertexes on a polygon.
<a href="#">options.useAutoPanInDrawing</a>	true	Type: Boolean  Enables autopan of the map when dragging a vertex on the boundary.
<a href="#">options.useMapMarginInDrawing</a>	true	Type: Boolean  Whether to use map margins in drawing mode.
<a href="#">options.vertexInteractiveOptions</a>	true	Type: Boolean  Allows the vertices placemarks to use options with postfixes linked to the current state of the vertex. The following postfixes are available: <ul style="list-style-type: none"> <li>• Hover - Options with this given postfix are used when the user hovers over a vertex with the mouse pointer.</li> <li>• Drag - Options with this given postfix are used when the user drags a vertex.</li> <li>• Active - Options with this given postfix are used when the context menu is opened for the vertex.</li> </ul> Examples of such options: <code>vertexLayoutHover</code> , <code>vertexIconImageSizeActive</code> , <code>vertexIconImageShapeHover</code> , etc. If you do not want to change the options of vertices placemarks depending on their state, then you will need to disable this option.
<a href="#">options.vertexLayout</a>	—	Type: Function  Class of the layout for placemarks on polygon vertexes.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>	The geometry being edited. Inherited from <a href="#">IGeometryEditor</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

Name	Type	Description
<a href="#">state</a>	<a href="#">IDataManager</a>	<p>Manager for the state of the geometry editor.</p> <p>Data fields that are available via the "get" and "set" methods:</p> <ul style="list-style-type: none"> <li>• <code>editing</code> - Checks whether editing mode is enabled. Type - Boolean. Default value - false.</li> <li>• <code>drawing</code> - Checks whether vertex drawing mode is enabled. Type - Boolean. Default value - false.</li> <li>• <code>drawingFrom</code> - Checks how new points are added in drawing mode. Accepts one of two string values: "begin" - points are added at the beginning of the polygon; "end" - points are added at the end.</li> <li>• <code>drawingFromIndex</code> - The index of the polygon vertex after which new points will be added in drawing mode. This field is available only during editing, because saving changes rearranges the order of vertexes in the polygon so that the point with the specified index becomes the last one. Type - integer.</li> <li>• <code>drawingPath</code> - The index of the polygon contour where new points are added in drawing mode. Type - integer. Default value - 0.</li> </ul>

## Events

Name	Description
<a href="#">beforeedgedrag</a>	<p>Event preceding the "edgedrag" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>pixelOffset</code> - Array of two numbers that describe the pixel offset at this step.</li> <li>• <code>edgeModel</code> - Reference to the model of the draggable placemark.</li> <li>• <code>globalPixels</code> - Coordinates of the draggable placemark, in global pixel coordinates.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>• <code>setPixelOffset</code> - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent "edgedrag" event will be canceled.</p>
<a href="#">beforeedgedragstart</a>	<p>Event preceding the "edgedragstart" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> <li>• <code>edgeModel</code> - Reference to the model of the draggable placemark.</li> <li>• <code>globalPixels</code> - Coordinates of the draggable placemark, in global pixel coordinates.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, any subsequent dragging, as well as the "edgedragstart" event, will be canceled.</p>

Name	Description
<a href="#">beforevertexadd</a>	<p>Event preceding the "vertexadd" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>parentModel - Reference to the parent data model for the vertex being added.</li> <li>vertexIndex - Index of the vertex being added.</li> <li>globalPixels - Coordinates of the vertex being added, in global pixel coordinates.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>setGlobalPixels - Use this method to correct the coordinate values of the added vertex. It takes an argument with the new global pixel coordinates of the vertex as an array of two numbers.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent "vertexadd" event will be canceled.</p>
<a href="#">beforevertexdrag</a>	<p>Event preceding the "vertexdrag" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>pixelOffset - Array of two numbers that describe the pixel offset at this step.</li> <li>vertexModel - Reference to the model of the draggable vertex.</li> <li>globalPixels - Coordinates of the draggable vertex, in global pixel coordinates.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>setPixelOffset - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent "vertexdrag" event will be canceled.</p>
<a href="#">beforevertexdragstart</a>	<p>Event preceding the "vertexdragstart" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> <li>vertexModel - Reference to the model of the draggable vertex.</li> <li>globalPixels - Coordinates of the draggable vertex, in global pixel coordinates.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, any subsequent dragging, as well as the "vertexdragstart" event, will be canceled.</p>
<a href="#">beforevertexdraw</a>	<p>Event preceding the "vertexdraw" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>parentModel - Reference to the parent data model for the vertex being added.</li> <li>vertexIndex - Index of the vertex being added.</li> <li>globalPixels - Coordinates of the vertex being added, in global pixel coordinates.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>setGlobalPixels - Use this method to correct the coordinate values of the added vertex. It takes an argument with the new global pixel coordinates of the vertex as an array of two numbers.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent "vertexdraw" event will be canceled.</p>
<a href="#">drawingstart</a>	Enabling the mode for adding new vertexes. Instance of the <a href="#">Event</a> class.
<a href="#">drawingstop</a>	Disabling the mode for adding new vertexes. Instance of the <a href="#">Event</a> class.

Name	Description
<a href="#">edgedrag</a>	<p>Dragging an interim placemark. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>pixelOffset - Array of two numbers that describe the pixel offset at this step.</li> <li>edgeModel - Reference to the model of the draggable placemark.</li> <li>globalPixels - Coordinates of the draggable placemark, in global pixel coordinates.</li> </ul>
<a href="#">edgedragend</a>	<p>End of dragging an interim placemark. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>edgeModel - Reference to the model of the draggable placemark.</li> <li>globalPixels - Coordinates of the draggable placemark, in global pixel coordinates.</li> </ul>
<a href="#">edgedragstart</a>	<p>Start of dragging an interim placemark. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> <li>edgeModel - Reference to the model of the draggable placemark.</li> <li>globalPixels - Coordinates of the draggable placemark, in global pixel coordinates.</li> </ul>
<a href="#">editingstart</a>	Enabling the mode for editing vertexes. Instance of the <a href="#">Event</a> class.
<a href="#">editingstop</a>	Disabling the mode for editing vertexes. Instance of the <a href="#">Event</a> class.
<a href="#">framingstart</a>	Enabling the zoom mode. Instance of the <a href="#">Event</a> class.
<a href="#">framingstop</a>	Disabling the zoom mode. Instance of the <a href="#">Event</a> class.
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">statechange</a>	<p>Change to the geometry editor state. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IGeometryEditor</a>.</p>
<a href="#">vertexadd</a>	<p>Adding a new vertex. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>parentModel - Reference to the parent data model for the vertex that was added.</li> <li>vertexIndex - Index of the vertex that was added.</li> <li>globalPixels - Coordinates of the vertex that was added, in global pixel coordinates.</li> </ul>
<a href="#">vertexdrag</a>	<p>Dragging a vertex. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>pixelOffset - Array of two numbers that describe the pixel offset at this step.</li> <li>vertexModel - Reference to the model of the draggable vertex.</li> <li>globalPixels - Coordinates of the draggable vertex, in global pixel coordinates.</li> </ul>
<a href="#">vertexdragend</a>	<p>End of vertex dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>vertexModel - Reference to the model of the draggable vertex.</li> <li>globalPixels - Coordinates of the draggable vertex, in global pixel coordinates.</li> </ul>

Name	Description
<a href="#">vertexdragstart</a>	<p>Start of vertex dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> <li>• <code>vertexModel</code> - Reference to the model of the draggable vertex.</li> <li>• <code>globalPixels</code> - Coordinates of the draggable vertex, in global pixel coordinates.</li> </ul>
<a href="#">vertexdraw</a>	<p>Drawing a new vertex. This event usually precedes the add vertex event, and occurs when the mouse moves and the mode for adding new vertexes is enabled. Based on data passed in this event, guide lines are displayed in vertex drawing mode. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>parentModel</code> - Reference to the parent data model for the vertex being added.</li> <li>• <code>vertexIndex</code> - Index of the vertex being added.</li> <li>• <code>globalPixels</code> - Coordinates of the vertex being added, in global pixel coordinates.</li> </ul>

## Methods

Name	Returns	Description
<a href="#">getModel()</a>	<a href="#">vow.Promise</a>	<p>Returns the promise object, which is confirmed by the model object at the time it is actually created, or is rejected with one of the following error messages:</p> <ul style="list-style-type: none"> <li>• Canceled - Editing mode is disabled until the model is actually created.</li> <li>• Editor wasn't started - Editing mode is not enabled.</li> </ul>
<a href="#">getModelSync()</a>	<a href="#">geometryEditor.model.RootPolygon</a> or null	<p>Returns the editor's model, or null if it is missing at the time of the call.</p>
<a href="#">getView()</a>	<a href="#">vow.Promise</a>	<p>Returns the promise object, which is confirmed by the representation object at the time it is actually created, or is rejected with one of the following error messages:</p> <ul style="list-style-type: none"> <li>• Canceled - Editing mode is disabled until the representation is actually created.</li> <li>• Editor wasn't started - Editing mode is not enabled.</li> </ul>
<a href="#">getViewSync()</a>	<a href="#">geometryEditor.view.MultiPath</a> or null	<p>Returns the editor's representation, or null if it is missing at the time of the call.</p>

Name	Returns	Description
<a href="#">startDrawing()</a>	<a href="#">vow.Promise</a>	Enables drawing mode (for adding new vertexes to a polygon). Enabling occurs asynchronously.
<a href="#">startEditing()</a>	<a href="#">vow.Promise</a>	Enables edit mode (for adding new vertexes to a polygon). Enabling occurs asynchronously.
<a href="#">startFraming()</a>	<a href="#">vow.Promise</a>	Enables the zoom mode for a polygon. Enabling occurs asynchronously.
<a href="#">stopDrawing()</a>		Disables drawing mode (for adding new vertexes to a polygon).
<a href="#">stopEditing()</a>		Disables edit mode.
<a href="#">stopFraming()</a>		Disables the zoom mode.

## Fields details

### state

```
{IDataManager} state
```

Manager for the state of the geometry editor.

Data fields that are available via the "get" and "set" methods:

- `editing` - Checks whether editing mode is enabled. Type - Boolean. Default value - false.
- `drawing` - Checks whether vertex drawing mode is enabled. Type - Boolean. Default value - false.
- `drawingFrom` - Checks how new points are added in drawing mode. Accepts one of two string values: "begin" - points are added at the beginning of the polygon; "end" - points are added at the end.
- `drawingFromIndex` - The index of the polygon vertex after which new points will be added in drawing mode. This field is available only during editing, because saving changes rearranges the order of vertexes in the polygon so that the point with the specified index becomes the last one. Type - integer.
- `drawingPath` - The index of the polygon contour where new points are added in drawing mode. Type - integer. Default value - 0.

## Events details

### beforeedgedrag

Event preceding the "edgedrag" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `pixelOffset` - Array of two numbers that describe the pixel offset at this step.
- `edgeModel` - Reference to the model of the draggable placemark.
- `globalPixels` - Coordinates of the draggable placemark, in global pixel coordinates.

Names of methods that are accessible via [Event.callMethod](#):

- `setPixelOffset` - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the [Event.preventDefault](#) method is called for this event, a subsequent "edgedrag" event will be canceled.

### beforeedgedragstart

Event preceding the "edgedragstart" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.
- `edgeModel` - Reference to the model of the draggable placemark.
- `globalPixels` - Coordinates of the draggable placemark, in global pixel coordinates.

If the [Event.preventDefault](#) method is called for this event, any subsequent dragging, as well as the "edgedragstart" event, will be canceled.

### beforevertexadd

Event preceding the "vertexadd" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `parentModel` - Reference to the parent data model for the vertex being added.
- `vertexIndex` - Index of the vertex being added.
- `globalPixels` - Coordinates of the vertex being added, in global pixel coordinates.

Names of methods that are accessible via [Event.callMethod](#):

- `setGlobalPixels` - Use this method to correct the coordinate values of the added vertex. It takes an argument with the new global pixel coordinates of the vertex as an array of two numbers.

If the [Event.preventDefault](#) method is called for this event, a subsequent "vertexadd" event will be canceled.

### Examples:

#### 1.

```
// Prohibiting adding new vertexes further than 100 pixels from the map center.
polygon.editor.events.add("beforevertexadd", function (event) {
  var mapGlobalPixelCenter = geoMap.getGlobalPixelCenter();
  var globalPixels = event.get("globalPixels");
  var vector = [mapGlobalPixelCenter[0] - globalPixels[0], mapGlobalPixelCenter[1] - globalPixels[1]];
  var vectorLength = Math.sqrt(vector[0] * vector[0] + vector[1] * vector[1]);
  if (dist > 100) {
    event.preventDefault();
  }
});
```

#### 2.

```
// Adjusting coordinates of the add vertex events so they fall inside a square
// with 100-pixel sides that is centered at the map center.
polygon.editor.events.add(["beforevertexdraw", "beforevertexadd"], function (event) {
  var mapGlobalPixelCenter = geoMap.getGlobalPixelCenter();
  var globalPixels = event.get("globalPixels");
  var pixelBounds = [
    [mapGlobalPixelCenter[0] - 100, mapGlobalPixelCenter[1] - 100],
    [mapGlobalPixelCenter[0] + 100, mapGlobalPixelCenter[1] + 100]
  ];
  event.callMethod("setGlobalPixels", [
    Math.max(Math.min(globalPixels[0], pixelBounds[1][0]), pixelBounds[0][0]),
    Math.max(Math.min(globalPixels[1], pixelBounds[1][1]), pixelBounds[0][1])
  ]);
});
```

### beforevertexdrag

Event preceding the "vertexdrag" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `pixelOffset` - Array of two numbers that describe the pixel offset at this step.
- `vertexModel` - Reference to the model of the draggable vertex.
- `globalPixels` - Coordinates of the draggable vertex, in global pixel coordinates.

Names of methods that are accessible via [Event.callMethod](#):

- `setPixelOffset` - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the [Event.preventDefault](#) method is called for this event, a subsequent "vertexdrag" event will be canceled.

### beforevertexdragstart

Event preceding the "vertexdragstart" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- domEvent - Source DOM event (as a [DomEvent](#) object), if there is one.
- vertexModel - Reference to the model of the draggable vertex.
- globalPixels - Coordinates of the draggable vertex, in global pixel coordinates.

If the [Event.preventDefault](#) method is called for this event, any subsequent dragging, as well as the "vertexdragstart" event, will be canceled.

### Example:

```
// Prohibiting dragging vertexes with the index 0 in a contour with the same index.
polygon.editor.events.add("beforevertexdragstart", function (event) {
  var vertexModel = event.get("vertexModel");
  var pathModel = vertexModel.getParent();
  if (pathModel.getIndex() == 0 && vertexModel.getIndex() == 0) {
    event.preventDefault();
  }
});
```

### beforevertexdraw

Event preceding the "vertexdraw" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- parentModel - Reference to the parent data model for the vertex being added.
- vertexIndex - Index of the vertex being added.
- globalPixels - Coordinates of the vertex being added, in global pixel coordinates.

Names of methods that are accessible via [Event.callMethod](#):

- setGlobalPixels - Use this method to correct the coordinate values of the added vertex. It takes an argument with the new global pixel coordinates of the vertex as an array of two numbers.

If the [Event.preventDefault](#) method is called for this event, a subsequent "vertexdraw" event will be canceled.

### drawingstart

Enabling the mode for adding new vertexes. Instance of the [Event](#) class.

### drawingstop

Disabling the mode for adding new vertexes. Instance of the [Event](#) class.

### edgedrag

Dragging an interim placemark. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- pixelOffset - Array of two numbers that describe the pixel offset at this step.
- edgeModel - Reference to the model of the draggable placemark.
- globalPixels - Coordinates of the draggable placemark, in global pixel coordinates.

### edgedragend

End of dragging an interim placemark. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- edgeModel - Reference to the model of the draggable placemark.
- globalPixels - Coordinates of the draggable placemark, in global pixel coordinates.

**edgedragstart**

Start of dragging an interim placemark. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.
- `edgeModel` - Reference to the model of the draggable placemark.
- `globalPixels` - Coordinates of the draggable placemark, in global pixel coordinates.

**editingstart**

Enabling the mode for editing vertexes. Instance of the [Event](#) class.

**editingstop**

Disabling the mode for editing vertexes. Instance of the [Event](#) class.

**framingstart**

Enabling the zoom mode. Instance of the [Event](#) class.

**framingstop**

Disabling the zoom mode. Instance of the [Event](#) class.

**vertexadd**

Adding a new vertex. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `parentModel` - Reference to the parent data model for the vertex that was added.
- `vertexIndex` - Index of the vertex that was added.
- `globalPixels` - Coordinates of the vertex that was added, in global pixel coordinates.

**vertexdrag**

Dragging a vertex. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `pixelOffset` - Array of two numbers that describe the pixel offset at this step.
- `vertexModel` - Reference to the model of the draggable vertex.
- `globalPixels` - Coordinates of the draggable vertex, in global pixel coordinates.

**vertexdragend**

End of vertex dragging. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `vertexModel` - Reference to the model of the draggable vertex.
- `globalPixels` - Coordinates of the draggable vertex, in global pixel coordinates.

**vertexdragstart**

Start of vertex dragging. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.
- `vertexModel` - Reference to the model of the draggable vertex.
- `globalPixels` - Coordinates of the draggable vertex, in global pixel coordinates.

**vertexdraw**

Drawing a new vertex. This event usually precedes the add vertex event, and occurs when the mouse moves and the mode for adding new vertexes is enabled. Based on data passed in this event, guide lines are displayed in vertex drawing mode. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `parentModel` - Reference to the parent data model for the vertex being added.

- `vertexIndex` - Index of the vertex being added.
- `globalPixels` - Coordinates of the vertex being added, in global pixel coordinates.

## Methods details

### getModel

```
{vow.Promise} getModel()
```

**Returns** the promise object, which is confirmed by the model object at the time it is actually created, or is rejected with one of the following error messages:

- Canceled - Editing mode is disabled until the model is actually created.
- Editor wasn't started - Editing mode is not enabled.

### getModelSync

```
{geometryEditor.model.RootPolygon|null} getModelSync()
```

**Returns** the editor's model, or null if it is missing at the time of the call.

### getView

```
{vow.Promise} getView()
```

**Returns** the promise object, which is confirmed by the representation object at the time it is actually created, or is rejected with one of the following error messages:

- Canceled - Editing mode is disabled until the representation is actually created.
- Editor wasn't started - Editing mode is not enabled.

### getViewSync

```
{geometryEditor.view.MultiPath|null} getViewSync()
```

**Returns** the editor's representation, or null if it is missing at the time of the call.

### startDrawing

```
{vow.Promise} startDrawing()
```

Enables drawing mode (for adding new vertexes to a polygon). Enabling occurs asynchronously.

**Returns** the promise object, which is confirmed when drawing mode has actually started, or is rejected with one of the following error messages:

- Canceled - Drawing mode is disabled until it is actually started.

### startEditing

```
{vow.Promise} startEditing()
```

Enables edit mode (for adding new vertexes to a polygon). Enabling occurs asynchronously.

**Returns** the promise object, which is confirmed when editing mode has actually started, or is rejected with one of the following error messages:

- Canceled - Editing mode is disabled until it is actually started.

### startFraming

```
{vow.Promise} startFraming()
```

Enables the zoom mode for a polygon. Enabling occurs asynchronously.

**Returns** the promise object that is confirmed when zoom mode actually starts.

### stopDrawing

```
{ } stopDrawing()
```

Disables drawing mode (for adding new vertexes to a polygon).

### stopEditing

```
{ } stopEditing()
```

Disables edit mode.

### stopFraming

```
{ } stopFraming()
```

Disables the zoom mode.

## geometryEditor.view

### geometryEditor.view.Edge

**Note:** The constructor of the geometryEditor.view.Edge class is hidden, as this class is not intended for autonomous initialization.

View of an interim placemark. The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand.

#### Methods

#### Methods

Name	Returns	Description
<a href="#">getPlacemark()</a>	<a href="#">GeoObject</a>	Returns a reference to the interim placemark geo object. The placemark data contains a reference to the data model for the interim placemark (called "model").

#### Methods details

### getPlacemark

```
{GeoObject} getPlacemark()
```

**Returns** a reference to the interim placemark geo object. The placemark data contains a reference to the data model for the interim placemark (called "model").

### geometryEditor.view.MultiPath

**Note:** The constructor of the geometryEditor.view.MultiPath class is hidden, as this class is not intended for autonomous initialization.

View of a set of contours. The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand.

#### Methods

**Methods**

Name	Returns	Description
<a href="#">getEdgePlacemarks()</a>	<a href="#">GeoObjectCollection</a>	Returns a collection of interim placemarks of child contour views.
<a href="#">getPathViews()</a>	<a href="#">geometryEditor.view.Path[]</a>	Returns an array of views of child contours.
<a href="#">getVertexPlacemarks()</a>	<a href="#">GeoObjectCollection</a>	Returns a collection of vertex placemarks of child contour views.

**Methods details****getEdgePlacemarks**

```
{GeoObjectCollection} getEdgePlacemarks()
```

**Returns** a collection of interim placemarks of child contour views.

**getPathViews**

```
{geometryEditor.view.Path[]} getPathViews()
```

**Returns** an array of views of child contours.

**getVertexPlacemarks**

```
{GeoObjectCollection} getVertexPlacemarks()
```

**Returns** a collection of vertex placemarks of child contour views.

**geometryEditor.view.Path**

**Note:** The constructor of the `geometryEditor.view.Path` class is hidden, as this class is not intended for autonomous initialization.

Contour view. The constructor is not available in the `package.full` (a standard set of modules). This module is loaded on demand.

[Methods](#)**Methods**

Name	Returns	Description
<a href="#">getEdgePlacemarks()</a>	<a href="#">GeoObjectCollection</a>	Returns a collection of interim placemarks that are on the contour.
<a href="#">getEdgeViews()</a>	<a href="#">geometryEditor.view.Edge[]</a>	Returns an array of views of interim placemarks.
<a href="#">getVertexPlacemarks()</a>	<a href="#">GeoObjectCollection</a>	Returns a collection of vertex placemarks that are on the contour.

Name	Returns	Description
<code>getVertexViews()</code>	<code>geometryEditor.view.Vertex[]</code>	Returns an array of views of child vertexes.

### Methods details

#### getEdgePlacemarks

```
{GeoObjectCollection} getEdgePlacemarks()
```

Returns a collection of interim placemarks that are on the contour.

#### getEdgeViews

```
{geometryEditor.view.Edge[]} getEdgeViews()
```

Returns an array of views of interim placemarks.

#### getVertexPlacemarks

```
{GeoObjectCollection} getVertexPlacemarks()
```

Returns a collection of vertex placemarks that are on the contour.

#### getVertexViews

```
{geometryEditor.view.Vertex[]} getVertexViews()
```

Returns an array of views of child vertexes.

#### geometryEditor.view.Vertex

**Note:** The constructor of the `geometryEditor.view.Vertex` class is hidden, as this class is not intended for autonomous initialization.

Vertex view. The constructor is not available in the `package.full` (a standard set of modules). This module is loaded on demand.

#### Methods

### Methods

Name	Returns	Description
<code>getPlacemark()</code>	<code>GeoObject</code>	Returns geo object of the placemark that marks the endpoint. The placemark data contains a reference to the data model for the vertex (called "model").

### Methods details

#### getPlacemark

```
{GeoObject} getPlacemark()
```

Returns geo object of the placemark that marks the endpoint. The placemark data contains a reference to the data model for the vertex (called "model").

## geoObject

### geoObject.addon

#### geoObject.addon.balloon

**Note:** The constructor of the `geoObject.addon.balloon` class is hidden, as this class is not intended for autonomous initialization.

Static object.

The module that makes it possible to use a balloon for a geo object. Adds the [IBalloonOwner](#) interface to the geo object ([GeoObject](#)). When enabling `package.full` (the standard set of modules), it is available by default. If [GeoObject](#) is enabled separately, this module must be explicitly specified in the loader. If [geoObject.addon.balloon](#) is enabled separately after creating [GeoObject](#), the [IBalloonOwner](#) interface won't be added. Then in order to initialize the balloon manager, you will need to use the `geoObject.addon.balloon#get` method.

[Methods](#)

#### Methods

Name	Returns	Description
<a href="#">get(geoObject)</a>	<a href="#">IPopupManager</a>	Returns geo object balloon manager.

#### Methods details

##### get

```
{IPopupManager} get(geoObject)
```

**Returns** geo object balloon manager.

##### Parameters:

Parameter	Default value	Description
<a href="#">geoObject</a> *	—	Type: <a href="#">IGeoObject</a> Geo object

\* Mandatory parameter/option.

##### Example:

```
ymaps.geoObject.addon.balloon.get(geoObject)
```

#### geoObject.addon.editor

**Note:** The constructor of the `geoObject.addon.editor` class is hidden, as this class is not intended for autonomous initialization.

Static object.

The module that makes it possible to use the editor for a geo object. Adds the `GeoObject#editor` field to [GeoObject](#). When enabling `package.full` (the standard set of modules), it is available by default. If [GeoObject](#) is enabled separately, this module must be explicitly specified in the loader. If [geoObject.addon.editor](#) is enabled separately after creating [GeoObject](#), the `GeoObject#editor` field won't be added. Then you will need to use the `geoObject.addon.editor#get` method for the editor.

[Methods](#)

**Methods**

Name	Returns	Description
<a href="#">create(geoObject)</a>		A function that creates the "editor" field for the passed geo object.
<a href="#">get(geoObject)</a>	<a href="#">IGeometryEditor</a>	Returns geo object editor.

**Methods details****create**

```
{ } create(geoObject)
```

A function that creates the "editor" field for the passed geo object.

**Parameters:**

Parameter	Default value	Description
<a href="#">geoObject</a> *	—	Type: <a href="#">IGeoObject</a> Geo object.

\* Mandatory parameter/option.

**get**

```
{IGeometryEditor} get(geoObject)
```

Returns geo object editor.

**Parameters:**

Parameter	Default value	Description
<a href="#">geoObject</a> *	—	Type: <a href="#">IGeoObject</a> Geo object

\* Mandatory parameter/option.

**Example:**

```
ymaps.geoObject.addon.editor.get(geoObject)
```

**geoObject.addon.hint**

**Note:** The constructor of the `geoObject.addon.hint` class is hidden, as this class is not intended for autonomous initialization.

Static object.

The module that makes it possible to use a hint for a geo object. Adds the [IHintOwner](#) interface to the geo object ([GeoObject](#)). When enabling `package.full` (the standard set of modules), it is available by default. If [GeoObject](#) is enabled separately, this module must be explicitly specified in the loader. If [geoObject.addon.hint](#) is enabled separately after creating [GeoObject](#), the [IHintOwner](#) interface won't be added. Then in order to initialize the balloon manager, you will need to use the `geoObject.addon.hint#get` method.

[Methods](#)

**Methods**

Name	Returns	Description
<a href="#">get(geoObject)</a>	<a href="#">IPopupManager</a>	Returns geo object hint manager.

**Methods details****get**

```
{IPopupManager} get (geoObject)
```

Returns geo object hint manager.

**Parameters:**

Parameter	Default value	Description
<a href="#">geoObject</a> *	—	Type: <a href="#">IGeoObject</a> Geo object

\* Mandatory parameter/option.

**Example:**

```
ymaps.geoObject.addon.hint.get (geoObject)
```

**geoObject.Balloon**

Extends [IBalloonManager](#).

Geo object balloon manager. Allows a geo object to manage a balloon by opening it and hiding it. Passes data to the balloon in the [IGeoObjectPopupData](#). Uses the map balloon manager [map.Balloon](#) internally. Geo objects contain an instance of this class, which is available as `myGeoObject.balloon`. Don't create new instances of this class unless necessary.

See [Balloon GeoObject.balloon](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
geoObject.Balloon (geoObject)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">geoObject</a> *	—	Type: Object Geo object.

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

## Events

Name	Description
<a href="#">autopanbegin</a>	<p>Start of automatic shifting of the map center initiated by the autoPan method. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>target - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> <p>Inherited from <a href="#">IBalloonManager</a>.</p>
<a href="#">autopanend</a>	<p>End of automatic shifting of the map center initiated by the autoPan method. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>target - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> <p>Inherited from <a href="#">IBalloonManager</a>.</p>
<a href="#">beforeuserclose</a>	<p>The event which precedes <a href="#">Balloon.event:userclose</a>. Allows you to cancel the user's action by calling the preventDefault method. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>target - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> <p>Inherited from <a href="#">IBalloonManager</a>.</p>
<a href="#">close</a>	<p>Closing the info object. Names of fields available via <a href="#">Event.get</a>:</p> <ul style="list-style-type: none"> <li>target - Reference to the object where the closing occurred.</li> </ul> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">open</a>	<p>Opening the info object. Names of fields available via <a href="#">Event.get</a>:</p> <ul style="list-style-type: none"> <li>target - Reference to the object where the opening occurred.</li> </ul> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">userclose</a>	<p>Balloon closed by the user. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>target - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> <p>Inherited from <a href="#">IBalloonManager</a>.</p>

## Methods

Name	Returns	Description
<a href="#">autoPan()</a>	<a href="#">vow.Promise</a>	<p>Moves the map so that the balloon is visible.</p> <p>Inherited from <a href="#">IBalloonManager</a>.</p>
<a href="#">close([force])</a>	<a href="#">vow.Promise</a>	<p>Closes the info object.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">destroy()</a>		<p>Disables the info object manager.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>

Name	Returns	Description
<a href="#">getData()</a>	Object null	Returns the data of the info object or 'null'.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOptions()</a>	<a href="#">IOptionManager</a>  null	Returns the options manager or 'null'.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object to return the overlay.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	Returns the overlay, if one exists.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">getPosition()</a>	Number[] null	Returns the coordinates of the info object or 'null'.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">isOpen()</a>	Boolean	Returns the info object state: open/closed.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">open([position[, data[, options]])</a>	<a href="#">vow.Promise</a>	Opens the balloon for the geo object.
<a href="#">setData([data])</a>	<a href="#">vow.Promise</a>	Sets new user data.
<a href="#">setOptions(options)</a>	<a href="#">vow.Promise</a>	Defines new options for the info object.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">setPosition(position)</a>	<a href="#">vow.Promise</a>	Specifies a new position for the info object.  Inherited from <a href="#">IPopupManager</a> .

### Methods details

#### open

```
{vow.Promise} open([position[, data[, options]])
```

Opens the balloon for the geo object.

**Returns** Promise object.

**Parameters:**

Parameter	Default value	Description
<a href="#">position</a>	—	Type: Number[]  Coordinates of the point where the hint is opened. By default: the point on the geoobject that is closest to the current center of the map. The projection of the coordinates can be specified in the options, otherwise the geo object projection is used.
<a href="#">data</a>	—	Type: Object  Data to add to the <i>userData</i> field for the object of data passed to the balloon.
<a href="#">options</a>	—	Type: Object  Options.

**setData**

```
{vow.Promise} setData([data])
```

Sets new user data.

**Returns** Promise object.

**Parameters:**

Parameter	Default value	Description
<a href="#">data</a>	—	Type: Object  Data to add to the <i>userData</i> field for the object of data passed to the balloon.

**geoObject.Hint**

Extends [IHintManager](#).

Geo object hint manager. Allows a geo object to manage a hint by opening it and hiding it. Passes data to the hint in the format [IGeoObjectPopupData](#). Uses the map hint manager [map.Hint](#) internally. Geo objects contain an instance of this class, which is available as `myGeoObject.hint`. Don't create new instances of this class unless necessary.

**See** [Hint GeoObject.hint](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
geoObject.Hint (geoObject)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">geoObject</a> *	—	Type: Object Geo object.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

## Events

Name	Description
<a href="#">close</a>	Closing the info object. Names of fields available via <a href="#">Event.get</a> : <ul style="list-style-type: none"> <li><code>target</code> - Reference to the object where the closing occurred.</li> </ul> Inherited from <a href="#">IPopupManager</a> .
<a href="#">open</a>	Opening the info object. Names of fields available via <a href="#">Event.get</a> : <ul style="list-style-type: none"> <li><code>target</code> - Reference to the object where the opening occurred.</li> </ul> Inherited from <a href="#">IPopupManager</a> .

## Methods

Name	Returns	Description
<a href="#">close</a> ( <a href="#">[force]</a> )	<a href="#">vow.Promise</a>	Closes the info object. Inherited from <a href="#">IPopupManager</a> .
<a href="#">destroy</a> ()		Disables the info object manager. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getData</a> ()	Object null	Returns the data of the info object or 'null'. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOptions</a> ()	<a href="#">IOptionManager</a>  null	Returns the options manager or 'null'. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOverlay</a> ()	<a href="#">vow.Promise</a>	Returns the promise object to return the overlay. Inherited from <a href="#">IPopupManager</a> .

Name	Returns	Description
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	Returns the overlay, if one exists.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">getPosition()</a>	Number[] null	Returns the coordinates of the info object or 'null'.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">isOpen()</a>	Boolean	Returns the info object state: open/closed.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">open([position[, data[, options]])</a>	<a href="#">vow.Promise</a>	Opens the hint for the geo object.
<a href="#">setData([data])</a>	<a href="#">vow.Promise</a>	Sets new user data.
<a href="#">setOptions(options)</a>	<a href="#">vow.Promise</a>	Defines new options for the info object.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">setPosition(position)</a>	<a href="#">vow.Promise</a>	Specifies a new position for the info object.  Inherited from <a href="#">IPopupManager</a> .

## Methods details

### open

```
{vow.Promise} open([position[, data[, options]])
```

Opens the hint for the geo object.

**Returns** Promise object.

#### Parameters:

Parameter	Default value	Description
<a href="#">position</a>	—	Type: Number[]  Coordinates of the point where the hint is opened. By default: the geometric center of gravity of the geoobject. The projection of the coordinates can be specified in the options, otherwise the geo object projection is used.

Parameter	Default value	Description
<a href="#">data</a>	—	Type: Object  Data to add to the <i>userData</i> field for the object of data passed to the hint.
<a href="#">options</a>	—	Type: Object  Options.

**setData**

```
{vow.Promise} setData([data])
```

Sets new user data.

**Returns** Promise object.

**Parameters:**

Parameter	Default value	Description
<a href="#">data</a>	—	Type: Object  Data to add to the <i>userData</i> field for the object of data passed to the hint.

**geoObject.Sequence**

Extends [IGeoObject](#), [IGeoObjectSequence](#).

An unchanged collection of geo objects. Allows you to group geo objects for adding them to the map, setting options, etc. Collections are geo objects too.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
geoObject.Sequence([feature[, options]])
```

Creates an unchangeable collection of geo objects.

**Parameters:**

Parameter	Default value	Description
<a href="#">feature</a>	—	Type: Object  Collection description. For all purposes, corresponds to the geo object description. See <a href="#">GeoObject</a> object.
<a href="#">feature.children</a>	—	Type: <a href="#">IGeoObject</a> []  Array of child geoobjects.
<a href="#">feature.geometry</a>	—	Type: <a href="#">IGeometry</a>  Object  Geometry of a collection.

Parameter	Default value	Description
<a href="#">feature.properties</a>	—	Type: <a href="#">IDataManager</a>  Object  Collection data.
<a href="#">options</a>	—	Type: Object  Collection options. You can set all the options described in the <a href="#">GeoObject</a> object. Option values will be applied both to the collection itself and to its child objects, if these options are not set for them.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">IDataManager</a>	Geo object data. Inherited from <a href="#">IGeoObject</a> .
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

## Events

Name	Description
<a href="#">boundschange</a>	Change to coordinates of the geographical area that spans the collection and its child geo objects. Instance of the <a href="#">Event</a> class.  Inherited from <a href="#">IGeoObjectSequence</a> .
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li></ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>oldMap - Old map.</li><li>newMap - New map.</li></ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>clientX - X coordinate of the touch relative to the viewable area of the browser.</li><li>clientY - Y coordinate of the touch relative to the viewable area of the browser.</li><li>pageX - X coordinate of the touch relative to the beginning of the document.</li><li>pageY - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li><code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li><code>oldParent</code> - Old parent.</li> <li><code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">pixelboundschange</a>	<p>Change to pixel coordinates of the area that includes the collection and its child geo objects. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IGeoObjectSequence</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">each(callback[, context])</a>		<p>Calls a handler function for each child geo object.</p> <p>Inherited from <a href="#">IGeoObjectSequence</a>.</p>
<a href="#">get(index)</a>	<a href="#">IGeoObject</a>	<p>Returns a child geo object with the specified index.</p> <p>Inherited from <a href="#">IGeoObjectSequence</a>.</p>

Name	Returns	Description
<a href="#">getBounds()</a>	Number[][] null	Returns geographical coordinates of the area that covers the collection and its child geo objects.  Inherited from <a href="#">IGeoObjectSequence</a> .
<a href="#">getIterator()</a>	<a href="#">IIterator</a>	Returns iterator for child geo objects in the collection.  Inherited from <a href="#">IGeoObjectSequence</a> .
<a href="#">getLength()</a>	Integer	Returns length of the collection.  Inherited from <a href="#">IGeoObjectSequence</a> .
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map.  Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.  Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay.  Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IChildOnMap</a> .
<a href="#">getPixelBounds()</a>	Number[][] null	Returns global pixel coordinates of the area that spans the collection and its child geo objects.  Inherited from <a href="#">IGeoObjectSequence</a> .
<a href="#">indexOf(object)</a>	Integer	Returns index of the child geo object. If the geo object cannot be found in the collection, -1 is returned.  Inherited from <a href="#">IGeoObjectSequence</a> .

Name	Returns	Description
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	<p>Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>

## GeoObject

Extends [IGeoObject](#).

Geo object. Can be displayed as a placemark, polyline, polygon, etc., depending on the geometry type. You can also use auxiliary classes for simplified creation of geo objects with a specific geometry type.

See [Placemark](#) [Polyline](#) [Polygon](#) [Circle](#) [Rectangle](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
GeoObject([feature[, options]])
```

### Parameters:

Parameter	Default value	Description
<a href="#">feature</a>	—	<p>Type: Object</p> <p>Geo object description.</p>
<a href="#">feature.geometry</a>	—	<p>Type: <a href="#">IGeometry</a> <a href="#">GeometryJson</a></p> <p>Geo object geometry. Can be either set directly by the object, or in the form of an object of the JSON representation of the geometry. Examples of classes and objects that can represent a geometry - <a href="#">geometry.Point</a>, <a href="#">geometry.json.Point</a>.</p>

Parameter	Default value	Description
<a href="#">feature.properties</a>	—	<p>Type: <a href="#">IDataManager</a> Object</p> <p>Geo object data. Can be set as a class instance implementing the <a href="#">IDataManager</a> interface, or as a hash. When options are set to default values, the following data fields are interpreted by a geo object:</p> <ul style="list-style-type: none"> <li>• <code>iconContent</code> - Content of the geo object's icon.</li> <li>• <code>iconCaption</code> - Caption for the geo object's icon.</li> <li>• <code>hintContent</code> - Content of the geo object's popup hint.</li> <li>• <code>balloonContent</code> - Content of the geo object's balloon.</li> <li>• <code>balloonContentHeader</code> - Content of the geo object balloon title.</li> <li>• <code>balloonContentBody</code> - Content of the main part of the geo object's balloon.</li> <li>• <code>balloonContentFooter</code> - Content of the lower part of the geo object's balloon.</li> </ul> <p>The <code>balloonContent</code> field is a shortcut for the <code>balloonContentBody</code> field, but if they are both set simultaneously, <code>balloonContentBody</code> takes priority. You can also add your own custom fields to the geo object data and use them wherever possible. For example, in the placemark layout or balloon layout.</p>
<a href="#">options</a>	—	<p>Type: Object</p> <p>Geo object options. Using this parameter, you can set options for the geo object itself, as well as for its parts:</p> <ul style="list-style-type: none"> <li>• Options for the <a href="#">geo object balloon</a> with the <code>balloon</code> prefix.</li> <li>• Options for the <a href="#">geo object's popup hint</a> with the <code>hint</code> prefix.</li> <li>• Options for the geo object's geometry editor with the <code>editor</code> prefix. The type of editor and list of available options depends on the geo object's geometry type. See the descriptions of <a href="#">geometryEditor.LineString</a>, <a href="#">geometryEditor.Polygon</a>, and <a href="#">geometryEditor.Point</a>.</li> <li>• Geometry options can be set without a prefix. See the descriptions of the <a href="#">IGeometry</a> classes for geometries <a href="#">geometry.Point</a>, <a href="#">geometry.Polygon</a>, and others.</li> </ul>

Parameter	Default value	Description
<a href="#">options.circleOverlay</a>	"default#circle"	Type: String Function  Key identifier from <a href="#">overlay.storage</a> or the overlay class. The generator function accepts three parameters: <ul style="list-style-type: none"> <li>geometry: <a href="#">IPixelCircleGeometry</a> - The pixel geometry itself.</li> <li>data: Object - The overlay data.</li> <li>options: Object - The overlay options.</li> </ul> And returns <a href="#">vow.Promise</a> .
<a href="#">options.cursor</a>	"pointer"	Type: String  Type of cursor over a geo object.
<a href="#">options.draggable</a>	false	Type: Boolean  Checks whether the geo object can be dragged.
<a href="#">options.fill</a>	true	Type: Boolean  Whether the shape is filled.
<a href="#">options.fillColor</a>	"0066ff99"	Type: String  Fill color.
<a href="#">options.fillImageHref</a>	—	Type: String  Background image. When this option is enabled in <b>stretch</b> mode, the "fillColor" value is ignored.
<a href="#">options.fillMethod</a>	'stretch'	Type: String  Type of background fill. Accepts one of two values: <ul style="list-style-type: none"> <li>stretch - The background image stretches to fit the size of the overlay.</li> <li>tile - The background image is repeated, without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.</li> </ul>
<a href="#">options.fillOpacity</a>	1	Type: Number  Fill transparency.
<a href="#">options.hasBalloon</a>	true	Type: Boolean  Checks whether the geo object has the "balloon" field.

Parameter	Default value	Description
<a href="#">options.hasHint</a>	true	Type: Boolean  Checks whether the geo object has the "hint" field.
<a href="#">options.hideIconOnBalloonOpen</a>	true	Type: Boolean  Hide the icon when opening the balloon.
<a href="#">options.iconCaptionMaxWidth</a>	188	Type: Number  Maximum width of the placemark caption.
<a href="#">options.iconColor</a>	—	Type: String  Icon color. This option is used for standard icons in browsers that support SVG.
<a href="#">options.iconContentLayout</a>	—	Type: Function String  Layout for icon contents. (Type: constructor for an object with the ILayout interface, or its key in the storage).
<a href="#">options.iconContentOffset</a>	—	Type: Number[]  Pixel offset for the icon contents relative to the parent element. Used in the default#imageWithContent layout.
<a href="#">options.iconContentPadding</a>	—	Type: Number[]  Padding for the icon contents.
<a href="#">options.iconContentSize</a>	—	Type: Number[]  Size of contents. Used in the default#imageWithContent layout.
<a href="#">options.iconImageClipRect</a>	[[0, 0], [{imageWidth}, {imageHeight}]]	Type: Number[][]  A rectangular area (the upper-left and lower-right corners are specified) that will be cut from the source image file and scaled to icon size (for example, for using sprites). By default, the entire source picture is used.
<a href="#">options.iconImageHref</a>	—	Type: String  URL of the icon's graphic file. Used only in combination with layouts (iconLayout) 'default ' #image' and 'default#imageWithContent'.
<a href="#">options.iconImageOffset</a>	—	Type: Number[]  Pixel offset for the icon image inside the parent element.

Parameter	Default value	Description
<a href="#">options.iconImageShape</a>	—	Type: <a href="#">IShape</a>  null  The hotspot shape. If not set, the rectangular shape based on the size and offset of the icon will be calculated automatically. The coordinates of the figure geometry are counted from the anchor point. Used only in combination with layouts ( <a href="#">iconLayout</a> ) 'default' '#image' and 'default#imageWithContent'.
<a href="#">options.iconImageSize</a>	—	Type: Number[]  Size of the icon, in pixels.
<a href="#">options.iconLayout</a>	—	Type: Function String  Icon layout. (Type: constructor for an object with the <a href="#">ILayout</a> interface, or its key in the storage).
<a href="#">options.iconMaxHeight</a>	—	Type: Number  Maximum height of an icon with contents.
<a href="#">options.iconMaxWidth</a>	—	Type: Number  Maximum width of an icon with contents.
<a href="#">options.iconOffset</a>	[0, 0]	Type: Number[]  Pixel offset of the upper-left corner of the icon relative to the geographic anchor point of a placemark.
<a href="#">options.iconShadow</a>	false	Type: Boolean  Flag for whether the icon has a shadow.
<a href="#">options.iconShadowImageClipRect</a>	[[0, 0], [{imageWidth}, {imageHeight}]]	Type: Number[][]  A rectangular area (the upper-left and lower-right corners are specified) that will be cut from the source image file and scaled to shadow size (for example, for using sprites). By default, the entire source picture is used.
<a href="#">options.iconShadowImageHref</a>	—	Type: String  URL of the graphic file for the icon shadow. Used only in combination with layouts ( <a href="#">iconShadowLayout</a> ) 'default' '#image' and 'default#imageWithContent'.

Parameter	Default value	Description
<a href="#">options.iconShadowImageOffset</a>	—	Type: Number[]  Pixel offset for the icon shadow image inside the parent element.
<a href="#">options.iconShadowImageSize</a>	—	Type: Number[]  Size of the icon shadow.
<a href="#">options.iconShadowLayout</a>	—	Type: Function String  Layout for the icon shadow. (Type: constructor for an object with the ILayout interface, or its key in the storage).
<a href="#">options.iconShadowOffset</a>	—	Type: Number[]  The pixel offset of the icon shadow relative to its set position.
<a href="#">options.interactiveZIndex</a>	—	Type: Boolean  Enables to automatically modify z-index of the geo object depending on its state. By default, takes the true value for <a href="#">geometry.Point</a> and false for other geometries.
<a href="#">options.interactivityModel</a>	"default#geoObject"	Type: String  Interactivity model. Available keys and their values are listed in the description of <a href="#">interactivityModel.storage</a> .
<a href="#">options.lineStringOverlay</a>	"default#polyline"	Type: String Function  Key identifier from <a href="#">overlay.storage</a> or the overlay class. The generator function accepts three parameters: <ul style="list-style-type: none"> <li>• geometry: <a href="#">IPixelLineStringGeometry</a> - The pixel geometry itself.</li> <li>• data: <a href="#">IDataManager</a> or Object - Overlay data.</li> <li>• options: Object - The overlay options.</li> </ul> And returns <a href="#">vow.Promise</a> .
<a href="#">options.opacity</a>	1	Type: Number  Transparency.
<a href="#">options.openBalloonOnClick</a>	true	Type: Boolean  Checks whether to show the balloon when the geo object is clicked on.

Parameter	Default value	Description
<a href="#">options.openEmptyBalloon</a>	false	Type: Boolean  Checks whether to show an empty balloon when the geo object is clicked on.
<a href="#">options.openEmptyHint</a>	false	Type: Boolean  Checks whether to show an empty hint when the mouse pointer hovers over the geo object.
<a href="#">options.openHintOnHover</a>	true	Type: Boolean  Checks whether to show a hint when the mouse pointer hovers over the geo object.
<a href="#">options.outline</a>	true	Type: Boolean  Whether the shape has an outline.
<a href="#">options.pane</a>	—	Type: String  The key of the pane where the geo object overlay is placed. The default value is defined by the current overlay.
<a href="#">options.pointOverlay</a>	"default#placemark"	Type: String Function  Key identifier from <a href="#">overlay.storage</a> or the overlay class. The generator function accepts three parameters: <ul style="list-style-type: none"> <li>• geometry: <a href="#">IPixelPointGeometry</a> - The pixel geometry itself.</li> <li>• data: <a href="#">IDataManager</a> or Object - Overlay data.</li> <li>• options: Object - The overlay options.</li> </ul> And returns <a href="#">vow.Promise</a> .
<a href="#">options.polygonOverlay</a>	"default#polygon"	Type: String Function  Key identifier from <a href="#">overlay.storage</a> or the overlay class. The generator function accepts three parameters: <ul style="list-style-type: none"> <li>• geometry: <a href="#">IPixelPolygonGeometry</a> - The pixel geometry itself.</li> <li>• data: <a href="#">IDataManager</a> or Object - Overlay data.</li> <li>• options: Object - The overlay options.</li> </ul> And returns <a href="#">vow.Promise</a> .
<a href="#">options.preset</a>	—	Type: String  Key for the geo object's preset options. The list of keys is provided in the description of <a href="#">option.presetStorage</a> .

Parameter	Default value	Description
<a href="#">options.rectangleOverlay</a>	"default#rectangle"	<p>Type: String Function</p> <p>Key identifier from <a href="#">overlay.storage</a> or the overlay class. The generator function accepts three parameters:</p> <ul style="list-style-type: none"> <li>geometry: <a href="#">IPixelCircleGeometry</a> - The pixel geometry itself.</li> <li>data: <a href="#">IDataManager</a> or Object - Overlay data.</li> <li>options: Object - The overlay options.</li> </ul> <p>And returns <a href="#">vow.Promise</a>.</p>
<a href="#">options.setMapCursorInDragging</a>	false	<p>Type: Boolean</p> <p>true – when dragging an object, set the cursor to "grabbing" for both the object and the map; false – only use the "grabbing" cursor on the object.</p>
<a href="#">options.strokeColor</a>	"0066ffff"	<p>Type: String String[]</p> <p>Color of the line or outline. You can set multiple values for a multistroke outline.</p>
<a href="#">options.strokeOpacity</a>	1	<p>Type: Number Number[]</p> <p>Transparency of the line or outline. You can set multiple values for a multistroke outline.</p>
<a href="#">options.strokeStyle</a>	—	<p>Type: String Object String[] Object[]</p> <p>Style of the line or outline. You can set multiple values for a multistroke outline.</p>
<a href="#">options.strokeWidth</a>	1	<p>Type: Number Number[]</p> <p>Thickness of the line or outline. You can set multiple values for a multistroke outline.</p>
<a href="#">options.syncOverlayInit</a>	false	<p>Type: Boolean</p> <p>Enables synchronously adding an overlay to the map. By default, overlays are added to the map asynchronously to prevent the browser from hanging when adding a large number of geo objects. However, adding asynchronously does not allow accessing the overlay immediately after adding a geo object to the map.</p>
<a href="#">options.useMapMarginInDragging</a>	true	<p>Type: Boolean</p> <p>When an object is dragged to the edge of the map, the map center changes automatically. Whether to use map margins when automatically shifting the map center with <a href="#">map.margin.Manager</a>.</p>

Parameter	Default value	Description
<code>options.visible</code>	<code>true</code>	Type: Boolean  Checks geo object visibility.
<code>options.zIndex</code>	—	Type: Number  The z-index of a geo object in its normal state. Lowest priority.
<code>options.zIndexActive</code>	—	Type: Number  The z-index of a geo object with an open balloon. Highest priority.
<code>options.zIndexDrag</code>	—	Type: Number  The z-index of a geo object that is being dragged.
<code>options.zIndexHover</code>	—	Type: Number  The z-index of a geo object when the mouse pointer is hovering over it.

**Examples:****1.**

```
var myGeoObject = new ymaps.GeoObject({
  // Defining a "Polyline" type of geometry.
  geometry: {
    type: "LineString",
    coordinates: [
      [55.75, 37.61], [52.51, 13.38]
    ]
  },
  // Defining the geo object data.
  properties: {
    hintContent: "Moscow-Berlin"
  }
}, {
  // Enabling the display mode as geodesic curves.
  geodesic: true,
  // Setting the width to 5 pixels.
  strokeWidth: 5,
  // Setting the line color.
  strokeColor: "#F008"
});
// Adding the geo object to the map.
geoMap.geoObjects.add(myGeoObject);
```

**2.**

```
var myGeoObject = new ymaps.GeoObject({
  // Defining a "Point" geometry.
  geometry: {
    type: "Point",
    coordinates: [55.75, 37.61]
  },
  // Defining the geo object data.
  properties: {
    hintContent: "Moscow",
    balloonContentHeader: "Moscow",
    balloonContentBody: "Capital of Russia",
    population: 11848762
  }
}, {
  // Setting the preset for a placemark with a dot and no content.
  preset: "islands#redDotIcon",
  // Enabling dragging.
  draggable: true,
  // Overriding the layout for the content in the lower part of the balloon.
  balloonContentFooterLayout: ymaps.templateLayoutFactory.createClass(
    'population: {{ properties.population }}; coordinates: {{ geometry.coordinates }}'
  ),
  // Disabling the delay for closing the popup hint.
  hintCloseTimeout: null
});
```

```
});
// Adding the geo object to the map.
geoMap.geoObjects.add(myGeoObject);
```

## Fields

Name	Type	Description
<a href="#">balloon</a>	<a href="#">geoObject.Balloon</a>	Balloon for a geo object.
<a href="#">editor</a>	<a href="#">IGeometryEditor</a>	Editor for the geo object geometry.
<a href="#">events</a>	<a href="#">event.Manager</a>	Event manager.
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry.
<a href="#">hint</a>	<a href="#">geoObject.Hint</a>	Geo object hint.
<a href="#">options</a>	<a href="#">option.Manager</a>	Geo object options manager.
<a href="#">properties</a>	<a href="#">data.Manager</a>	Geo object data manager.
<a href="#">state</a>	<a href="#">data.Manager</a>	State of the geo object. Defined by the following fields: <ul style="list-style-type: none"> <li><code>active</code>: Boolean - Indicates that a balloon is open on the geo object.</li> <li><code>hover</code>: Boolean - Indicates that the mouse is currently pointed at the geo object.</li> <li><code>drag</code>: Boolean - Indicates that the geo object is being dragged</li> </ul>

## Events

Name	Description
<a href="#">balloonclose</a>	Closing the balloon. Instance of the <a href="#">Event</a> class.
<a href="#">balloonopen</a>	Opening a balloon on a geo object. Instance of the <a href="#">Event</a> class.
<a href="#">beforedrag</a>	<p>Event preceding the "drag" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>position</code> - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li><code>pixelOffset</code> - Array of two numbers that describe the pixel offset at this step.</li> <li><code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li><code>setPixelOffset</code> - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent drag event will be canceled.</p>
<a href="#">beforedragstart</a>	<p>Event preceding the "dragstart" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>position</code> - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li><code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled.</p>

Name	Description
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">drag</a>	<p>Dragging a geo object. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>position</code> - Coordinates relative to the document. Array in the format [pageX, pageY].</li><li>• <code>pixelOffset</code> - Array of two numbers that describe the pixel offset at this step.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>
<a href="#">dragend</a>	<p>End of geo object dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>position</code> - Coordinates relative to the document. Array in the format [pageX, pageY].</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>
<a href="#">dragstart</a>	<p>Start of geo object dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>position</code> - Coordinates relative to the document. Array in the format [pageX, pageY].</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>
<a href="#">editorstatechange</a>	<p>Change in the state of the editor for the geo object's geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>originalEvent</code> - Original event of the geometry editor.</li></ul>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the geometry.</li></ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">hintclose</a>	<p>Closing the hint. Instance of the <a href="#">Event</a> class.</p>
<a href="#">hintopen</a>	<p>Opening a hint on a geo object. Instance of the <a href="#">Event</a> class.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldMap</code> - Old map.</li><li>• <code>newMap</code> - New map.</li></ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>

Name	Description
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	<p>Returns reference to the map.</p> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	<p>Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	<p>The method provides synchronous access to the overlay.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	<p>Returns link to the parent object, or null if the parent element was not set.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	<p>Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>

## Fields details

### balloon

```
{geoObject.Balloon} balloon
```

Balloon for a geo object.

### editor

```
{IGeometryEditor} editor
```

Editor for the geo object geometry.

#### Example:

```
// Start editing the geo object geometry.
geoObject.editor.startEditing();
// ...
// Finish editing.
geoObject.editor.stopEditing();
```

### events

```
{event.Manager} events
```

Event manager.

### geometry

```
{IGeometry|null} geometry
```

Geo object geometry.

#### Example:

```
// When changing the coordinates of a geo object's geometry, we set the map borders
// so that the geo object is covered entirely.
myMap.geoObjects.add(myGeoObject);
myGeoObject.geometry.events.add("change", function () {
    myMap.setBounds(myGeoObject.geometry.getBounds());
});
```

### hint

```
{geoObject.Hint} hint
```

Geo object hint.

### options

```
{option.Manager} options
```

Geo object options manager.

### properties

```
{data.Manager} properties
```

Geo object data manager.

#### Example:

```
// When changing data, output a custom ID
// if the object's "synchronized" field is set to false.
```

```
myGeoObject.properties.events.add("change", function () {
  if (!myGeoObject.properties.get("synchronized")) {
    console.log(myGeoObject.properties.get("myID"));
  }
});
```

## state

```
{data.Manager} state
```

State of the geo object. Defined by the following fields:

- **active**: Boolean - Indicates that a balloon is open on the geo object.
- **hover**: Boolean - Indicates that the mouse is currently pointed at the geo object.
- **drag**: Boolean - Indicates that the geo object is being dragged

## Events details

### balloonclose

Closing the balloon. Instance of the [Event](#) class.

### balloonopen

Opening a balloon on a geo object. Instance of the [Event](#) class.

### beforedrag

Event preceding the "drag" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- **position** - Coordinates relative to the document. Array in the format [pageX, pageY].
- **pixelOffset** - Array of two numbers that describe the pixel offset at this step.
- **domEvent** - Source DOM event (as a [DomEvent](#) object), if there is one.

Names of methods that are accessible via [Event.callMethod](#):

- **setPixelOffset** - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the [Event.preventDefault](#) method is called for this event, a subsequent drag event will be canceled.

### Example:

```
// Allowing dragging the geo object only along the horizontal axis.
geoObject.events.add("beforedrag", function (event) {
  var originalOffset = event.get("pixelOffset");
  event.callMethod("setPixelOffset", [originalOffset[0], 0]);
});
```

### beforedragstart

Event preceding the "dragstart" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- **position** - Coordinates relative to the document. Array in the format [pageX, pageY].
- **domEvent** - Source DOM event (as a [DomEvent](#) object), if there is one.

If the [Event.preventDefault](#) method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled.

### drag

Dragging a geo object. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- **position** - Coordinates relative to the document. Array in the format [pageX, pageY].

- `pixelOffset` - Array of two numbers that describe the pixel offset at this step.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

**dragend**

End of geo object dragging. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `position` - Coordinates relative to the document. Array in the format [pageX, pageY].
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

**dragstart**

Start of geo object dragging. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `position` - Coordinates relative to the document. Array in the format [pageX, pageY].
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

**editorstatechange**

Change in the state of the editor for the geo object's geometry. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `originalEvent` - Original event of the geometry editor.

**hintclose**

Closing the hint. Instance of the [Event](#) class.

**hintopen**

Opening a hint on a geo object. Instance of the [Event](#) class.

## GeoObjectCollection

Extends [IGeoObject](#), [IGeoObjectCollection](#).

Collection of geo objects. Allows you to group geo objects for adding them to the map, setting options, etc. Collections are geo objects too.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
GeoObjectCollection([[feature[, options]])
```

Creates a collection of geo objects.

**Parameters:**

Parameter	Default value	Description
<a href="#">feature</a>	—	Type: Object  Collection description. For all purposes, corresponds to the geo object description. See <a href="#">GeoObject</a> object.
<a href="#">feature.children</a>	—	Type: <a href="#">IGeoObject[]</a>  Array of child geoobjects.

Parameter	Default value	Description
<a href="#">feature.geometry</a>	—	Type: <a href="#">IGeometry</a>  Object  Geometry of a collection.
<a href="#">feature.properties</a>	—	Type: <a href="#">IDataManager</a>  Object  Collection data.
<a href="#">options</a>	—	Type: Object  Collection options. You can set all the options described in the <a href="#">GeoObject</a> object. Option values will be applied both to the collection itself and to its child objects, if these options are not set for them.

**Example:**

```
// Creating a collection of geo objects and setting options.
var myGeoObjects = new ymaps.GeoObjectCollection({}, {
  preset: "islands#redCircleIcon",
  strokeWidth: 4,
  geodesic: true
});

// Adding placemarks and a polyline to the collection.
myGeoObjects.add(new ymaps.Placemark([13.38, 52.51]));
myGeoObjects.add(new ymaps.Placemark([30.30, 50.27]));
myGeoObjects.add(new ymaps.Polyline([[13.38, 52.51], [30.30, 50.27]]));

// Adding the collection to the map.
myMap.geoObjects.add(myGeoObjects);
// Setting the map center and scale so that the whole collection is visible.
myMap.setBounds(myGeoObjects.getBounds());
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry.  Inherited from <a href="#">IGeoObject</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager.  Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">IDataManager</a>	Geo object data.  Inherited from <a href="#">IGeoObject</a> .
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object.  Inherited from <a href="#">IGeoObject</a> .

## Events

Name	Description
<a href="#">add</a>	<p>A child geo object has been added (inserted). Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>index: Integer - Index of the added geo object.</li> <li>child: <a href="#">IGeoObject</a> - Reference to the added geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObjectCollection</a>.</p>
<a href="#">boundschange</a>	<p>Change to coordinates of the geographical area that includes the collection and all its child geo objects. Instance of the <a href="#">Event</a> class.</p>
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">pixelboundschange</a>	<p>Change to pixel coordinates of the area that includes the collection and all its child geo objects. Instance of the <a href="#">Event</a> class.</p>

Name	Description
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">remove</a>	<p>A child geo object has been removed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>index: Integer - Index of the deleted geo object.</li> <li>child: <a href="#">IGeoObject</a> - Reference to the deleted geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObjectCollection</a>.</p>
<a href="#">set</a>	<p>A new child geo object has been added to the collection. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>index: Integer - Index of the geo object.</li> <li>child: <a href="#">IGeoObject</a> - Reference to the new geo object.</li> <li>prevChild: <a href="#">IGeoObject</a> - Reference to the previous value for this index.</li> </ul> <p>Inherited from <a href="#">IGeoObjectCollection</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">add(child)</a>	<a href="#">GeoObjectCollection</a>	Adds a geo object to the collection.
<a href="#">each(callback[, context])</a>		Iterates through all the items in the collection and calls a handler function for each of them.
<a href="#">get(index)</a>	<a href="#">IGeoObject</a>	Returns a child geo object with the specified index.  Inherited from <a href="#">IGeoObjectCollection</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns geographical coordinates of the area that covers the collection and all its child geo objects.
<a href="#">getIterator()</a>	<a href="#">Iterator</a>	Returns iterator for the collection.
<a href="#">getLength()</a>	Integer	Returns the number of geo objects in the collection.
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map.  Inherited from <a href="#">IParentOnMap</a> .

Name	Returns	Description
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.  Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay.  Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IChildOnMap</a> .
<a href="#">getPixelBounds()</a>	Number[][] null	Returns global pixel coordinates of the area that spans the collection and all its child geo objects.
<a href="#">indexOf(object)</a>	Integer	Returns index of the child geo object. If the geo object cannot be found in the collection, -1 is returned.  Inherited from <a href="#">IGeoObjectCollection</a> .
<a href="#">remove(child)</a>	<a href="#">GeoObjectCollection</a>	Removes a geo object from the collection.
<a href="#">removeAll()</a>	<a href="#">GeoObjectCollection</a>	Removes all the geo objects from the collection.
<a href="#">set(index, child)</a>	<a href="#">GeoObjectCollection</a>	Adds a new child geo object to the collection.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .
<a href="#">splice(index, number)</a>	<a href="#">GeoObjectCollection</a>	Removes geo objects from the collection. If necessary, puts other objects in their place. Objects that will be added in place of the deleted ones are passed as additional parameters (after the "number" parameter).
<a href="#">toArray()</a>	<a href="#">IGeoObject[]</a>	Returns an array containing all the collection's geo objects at the time of the method call.

## Events details

### boundschange

Change to coordinates of the geographical area that includes the collection and all its child geo objects. Instance of the [Event](#) class.

### pixelboundschange

Change to pixel coordinates of the area that includes the collection and all its child geo objects. Instance of the [Event](#) class.

## Methods details

### add

```
{GeoObjectCollection} add(child)
```

Adds a geo object to the collection.

**Returns** a reference to the collection.

#### Parameters:

Parameter	Default value	Description
<a href="#">child</a> *	—	Type: <a href="#">IGeoObject</a> Child object.

\* Mandatory parameter/option.

### each

```
{ } each(callback[, context])
```

Iterates through all the items in the collection and calls a handler function for each of them.

#### Parameters:

Parameter	Default value	Description
<a href="#">callback</a> *	—	Type: Function Handler function.
<a href="#">context</a>	—	Type: Object Context for the function.

\* Mandatory parameter/option.

### getBounds

```
{Number[][]|null} getBounds()
```

**Returns** geographical coordinates of the area that covers the collection and all its child geo objects.

#### Example:

```
// Setting the map center and zoom so that the entire collection is displayed.  
myMap.setBounds(myCollection.getBounds());
```

**getIterator**

```
{Iterator} getIterator()
```

**Returns** iterator for the collection.

**Example:**

```
// Searching the collection for a geo object with the "Polyline" geometry.
var iterator = myGroup.getIterator(),
    object;
while ((object = iterator.getNext()) != iterator.STOP_ITERATION) {
    if (object.geometry.getType() == "LineString") {
        break;
    }
}
```

**getLength**

```
{Integer} getLength()
```

**Returns** the number of geo objects in the collection.

**getPixelBounds**

```
{Number[][]|null} getPixelBounds()
```

**Returns** global pixel coordinates of the area that spans the collection and all its child geo objects.

**remove**

```
{GeoObjectCollection} remove(child)
```

Removes a geo object from the collection.

**Returns** a reference to the collection.

**Parameters:**

Parameter	Default value	Description
<code>child</code> *	—	Type: <a href="#">IGeoObject</a> Child object.

\* Mandatory parameter/option.

**removeAll**

```
{GeoObjectCollection} removeAll()
```

Removes all the geo objects from the collection.

**Returns** a reference to the collection.

**set**

```
{GeoObjectCollection} set(index, child)
```

Adds a new child geo object to the collection.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>index *</code>	—	Type: Integer Index.
<code>child *</code>	—	Type: <a href="#">IGeoObject</a> Child object.

\* Mandatory parameter/option.

## splice

```
{GeoObjectCollection} splice(index, number)
```

Removes geo objects from the collection. If necessary, puts other objects in their place. Objects that will be added in place of the deleted ones are passed as additional parameters (after the "number" parameter).

**Returns** collection of deleted geo objects.

### Parameters:

Parameter	Default value	Description
<code>index *</code>	—	Type: Integer Index of the geo object to start deletion from.
<code>number *</code>	—	Type: Integer The number of geo objects to be deleted.

\* Mandatory parameter/option.

### Example:

```
// Removes the second object.
myGeoObjects.splice(1, 1);
// Puts a new "obj" object in the second position.
myGeoObjects.splice(1, 0, obj);
// Replaces the second object with the new "obj" object.
myGeoObjects.splice(1, 1, obj);
```

## toArray

```
{IGeoObject[]} toArray()
```

**Returns** an array containing all the collection's geo objects at the time of the method call.

## geoQuery

Static function.

Forms a data set from the specified source and creates an instance of `GeoQueryResult` based on it.

**Returns** result containing data from the source.

```
{ GeoQueryResult } geoQuery(source)
```

### Parameters:

Parameter	Default value	Description
<code>source</code> *	—	<p>Type: Object</p> <p>Source of geo objects:</p> <ul style="list-style-type: none"> <li>• <code>IGeoObject</code> - Object that implements the corresponding interface.</li> <li>• <code>IGeoObject[]</code> - Array of objects that implement the corresponding interface.</li> <li>• <code>ICollection</code> - Collection of objects that implement the <code>IGeoObject</code> interface.</li> <li>• <code>ICollection[]</code> - Array of object collections that implement the <code>IGeoObject</code> interface.</li> <li>• <code>vow.Promise</code> - A promise object that passes the data source for <code>geoQuery</code> to the handler function. The handler function can also be passed an object with the <code>geoObjects</code> field containing the data source for <code>geoQuery</code>.</li> <li>• <code>GeoQueryResult</code> - Object of the <code>GeoQueryResult</code> class.</li> <li>• <code>String Object</code> - String or object with a JSON description of objects.</li> </ul> <p>JSON object descriptions are formed using the following approach (see the example below). An object may be an entity or a collection of entities. A collection of entities is made up of an object with the following fields:</p> <ul style="list-style-type: none"> <li>• <code>type</code> - Type of object. The value of the field must be "FeatureCollection".</li> <li>• <code>features</code> - Array of child entities in the collection. Child objects may be entities or nested collections of entities.</li> </ul> <p>An entity is made up of an object with the following fields:</p> <ul style="list-style-type: none"> <li>• <code>type</code> - Type of object. The value of the field must be "Feature".</li> <li>• <code>geometry</code> - Object geometry. Contains the "type" and "coordinates" fields. Corresponds to the parameter passed to the constructor for the <code>ymaps.GeoObject</code> object.</li> <li>• <code>options</code> - Options for the geo object.</li> <li>• <code>properties</code> - Geo object data.</li> </ul>

\* Mandatory parameter/option.

### Examples:

#### 1.

```
// Creating GeoQueryResult from a single geo object.
var placemark = new ymaps.Placemark([34, 56]);
```

```
ymaps.geoQuery(placemark).addToMap(myMap);
```

## 2.

```
// Creating GeoQueryResult from an array of geo objects.
var objects = [
  new ymaps.Placemark([34, 56]),
  new ymaps.Rectangle([[34, 56], [36, 57]])
];
ymaps.geoQuery(objects).addToMap(myMap);
```

## 3.

```
// Creating GeoQueryResult from a collection of geo objects.
var result = ymaps.geoQuery(myMap.geoObjects).searchIntersect(myMap);
alert("Number of geo objects in the visible map area: " + result.getLength());
```

## 4.

```
// Creating GeoQueryResult from vow.Deferred.
var result = ymaps.geoQuery(ymaps.geocode('Siromyatnicheskiy lane')).searchInside(myGeoBounds);
// Because the data source is asynchronous, we must wait for result processing.
result.then(function () {
  alert('Number of objects located inside the specified area: ' + result.getLength());
});
```

## 5.

```
// Creating GeoQueryResult from JSON.
var result = ymaps.geoQuery({
  type: 'FeatureCollection',
  features: [
    {
      type: 'Feature',
      geometry: {
        type: 'Circle',
        coordinates: [15, 15],
        radius: 100
      }
    },
    {
      type: 'Feature',
      geometry: {
        type: 'LineString',
        coordinates: [[15, 16], [66, 23]]
      }
    },
    {
      type: 'FeatureCollection',
      features: [
        {
          type: 'Feature',
          geometry: {
            type: 'Point',
            coordinates: [12, 41]
          },
          properties: {
            name: 'point'
          },
          options: {
            preset: 'islands#yellowIcon'
          }
        }
      ]
    }
  ]
});
// Adding non-point objects to the map as-is.
result.search('geometry.type != "Point").addToMap(myMap);
// Adding point objects to the map via the clusterer.
myMap.geoObjects.add(result.search('geometry.type == "Point").clusterize());
```

## GeoQueryResult

Extends [IPromiseProvider](#).

Geo query result.

[Constructor](#) | [Methods](#)

**Constructor**

```
GeoQueryResult(source)
```

**Parameters:**

Parameter	Default value	Description
<code>source</code> *	—	<p>Type: Object</p> <p>Source of geo objects:</p> <ul style="list-style-type: none"> <li>IGeoObject - Object that implements the corresponding interface.</li> <li>IGeoObject[] - Array of objects that implement the corresponding interface.</li> <li>ICollection - Collection of objects that implement the IGeoObject interface.</li> <li>ICollection[] - Array of object collections that implement the IGeoObject interface.</li> <li>IPromiseProvider - A promise object that passes the data source for geoQuery to the handler function. The handler function can also be passed an object with the geoObjects field containing the data source for geoQuery.</li> <li>GeoQueryResult - Object of the GeoQueryResult class.</li> <li>String Object - String or object with a JSON description of objects.</li> <li>Object[] - Array of JSON descriptions of geometries. An array item is an object with the "type" and "coordinates" fields. When describing a circle, the "radius" field is also mandatory. When describing a polygon, the optional "fillRule" field may also be specified.</li> </ul> <p>JSON object descriptions are formed using the following approach (see the example below). An object may be an entity or a collection of entities. A collection of entities is made up of an object with the following fields:</p> <ul style="list-style-type: none"> <li>type - Type of object. The value of the field must be "FeatureCollection".</li> <li>features - Array of child entities in the collection. Child objects may be entities or nested collections of entities.</li> </ul> <p>An entity is made up of an object with the following fields:</p> <ul style="list-style-type: none"> <li>type - Type of object. The value of the field must be "Feature".</li> <li>geometry - Object geometry. Contains the "type" and "coordinates" fields. Corresponds to the parameter passed to the constructor for the ymaps.GeoObject object.</li> <li>options - Options for the geo object.</li> <li>properties - Geo object data.</li> </ul>

\* Mandatory parameter/option.

## Examples:

### 1.

```
// Creating GeoQueryResult from a single geo object.
var placemark = new ymaps.Placemark([34, 56]);
ymaps.geoQuery(placemark).addToMap(myMap);
```

### 2.

```
// Creating GeoQueryResult from an array of geo objects.
var objects = [
  new ymaps.Placemark([34, 56]),
  new ymaps.Rectangle([[34, 56], [36, 57]])
];
ymaps.geoQuery(objects).addToMap(myMap);
```

### 3.

```
// Creating GeoQueryResult from a collection of geo objects.
var result = ymaps.geoQuery(myMap.geoObjects).searchIntersect(myMap);
alert("Number of geo objects in the visible map area: " + result.getLength());
```

### 4.

```
// Creating GeoQueryResult from vow.Deferred.
var result = ymaps.geoQuery(ymaps.geocode('Siromyatnicheskiy lane')).searchInside(myGeoBounds);
// Because the data source is asynchronous, we must wait for result processing.
result.then(function () {
  alert('Number of objects located inside the specified area: ' + result.getLength());
});
```

### 5.

```
// Creating GeoQueryResult from JSON.
var result = ymaps.geoQuery({
  type: 'FeatureCollection',
  features: [
    {
      type: 'Feature',
      geometry: {
        type: 'Circle',
        coordinates: [15, 15],
        radius: 100
      }
    },
    {
      type: 'Feature',
      geometry: {
        type: 'LineString',
        coordinates: [[15, 16], [66, 23]]
      }
    }
  ],
  // Object collections can be nested.
  {
    type: 'FeatureCollection',
    features: [
      {
        type: 'Feature',
        geometry: {
          type: 'Point',
          coordinates: [12, 41]
        },
        properties: {
          name: 'point'
        },
        options: {
          preset: 'islands#yellowIcon'
        }
      }
    ]
  }
]
});
// Adding non-point objects to the map as-is.
result.search('geometry.type != "Point").addToMap(myMap);
// Adding point objects to the map via the clusterer.
myMap.geoObjects.add(result.search('geometry.type == "Point").clusterize());
```

**Methods**

<b>Name</b>	<b>Returns</b>	<b>Description</b>
<a href="#">add(source)</a>	<a href="#">GeoQueryResult</a>	Adds objects from the source to objects in the result. It does not change the source object; it creates a new object containing a combined set of geo objects.
<a href="#">addEvents(events, callback, context)</a>	<a href="#">GeoQueryResult</a>	Assigns event handlers to selection items.
<a href="#">addTo(collection)</a>	<a href="#">GeoQueryResult</a>	Method for adding objects to a collection of geo objects.
<a href="#">addToMap(map)</a>	<a href="#">GeoQueryResult</a>	Method for adding objects to the map.
<a href="#">applyBoundsToMap(map[, options])</a>	<a href="#">GeoQueryResult</a>	Method for setting the visible area of the map so that all the objects in the selection are visible.
<a href="#">clusterize([options])</a>	<a href="#">Clusterer</a>	This method creates a clusterer and adds objects from the selection to it. If the selection data are not ready yet, they will be added to the clusterer immediately after processing, and the returned clusterer will be empty at first. Only objects with the "Point" geometry will be added to the clusterer.
<a href="#">each(callback, context)</a>	<a href="#">GeoQueryResult</a>	Returns self-reference.
<a href="#">get(index)</a>	<a href="#">IGeoObject</a>	Returns a selection item by index.
<a href="#">getBounds()</a>	Number[][] null	Returns geographical coordinates of the area that includes all the result objects.
<a href="#">getCenter(map)</a>	Number[]	Method that returns the center of the area that covers all the result objects, in geographical coordinates.
<a href="#">getCentralObject(map)</a>	<a href="#">IGeoObject</a>  null	Method for getting the object that is closest to the center of the visible area of the map.

Name	Returns	Description
<a href="#">getClosestTo(object)</a>	<a href="#">IGeoObject</a>  null	Method that returns the selection object that is closest to the one specified. If an object that is already in the selection is given as input, it returns a different object from the selection that is closest to the one specified. Note that many of the geo objects must be added to the map for correct calculations.
<a href="#">getExtreme(key)</a>	Number	Returns the maximum and minimum coordinate values among the coordinates of objects in the selection.
<a href="#">getExtremeObject(key)</a>	<a href="#">IGeoObject</a>	Method that returns the object with the minimum or maximum coordinates among the coordinates of objects in the selection.
<a href="#">getGlobalPixelBounds(map)</a>	Number[][] null	Method that returns global pixel coordinates of the area for the current map zoom.
<a href="#">getGlobalPixelCenter(map)</a>	Number[]	Returns coordinates of the center of the area, that covers all the result objects, in global pixel coordinates for the current map zoom.
<a href="#">getIterator()</a>	<a href="#">Iterator</a>	Returns iterator for objects in the result.
<a href="#">getLength()</a>	Number	Returns number of items in the result.
<a href="#">getMaxZoom(map[, options])</a>	Number	Method that calculates the maximum zoom level at which all the objects fall within the visible area of the map.
<a href="#">getParent()</a>	<a href="#">GeoQueryResult</a>  null	Returns reference to the parent selection, if the current selection was created as the result of changing another <a href="#">GeoQueryResult</a> object.
<a href="#">indexOf(item)</a>	Number	Returns index of the item in the selection. If the item was not found, it returns -1.
<a href="#">intersect(result)</a>	<a href="#">GeoQueryResult</a>	This method creates a new selection containing common items for two different selections.

Name	Returns	Description
<a href="#">isReady()</a>	Boolean	Returns whether the selection results are ready or are still being processed.
<a href="#">map(callback[, context])</a>	<a href="#">GeoQueryResult</a>	Method that calls the "callback" method for all the selection items and forms a new selection based on the results received.
<a href="#">remove(objects)</a>	<a href="#">GeoQueryResult</a>	Deletes objects from the result. It does not change the source object; it creates a new object containing the resulting set of geo objects.
<a href="#">removeEvents(events, callback, context)</a>		Deletes the subscription to events from an object. Note that in order to unsubscribe correctly, the arguments passed must be exactly the same as the ones for subscribing via the "addEvents" method.
<a href="#">removeFrom(collection)</a>	<a href="#">GeoQueryResult</a>	Method for deleting objects from a collection.
<a href="#">removeFromMap(map)</a>	<a href="#">GeoQueryResult</a>	Method for deleting objects from the map.
<a href="#">reverse()</a>	<a href="#">GeoQueryResult</a>	Rearranges the selection items in reverse order and returns the new selection.
<a href="#">search(condition)</a>	<a href="#">GeoQueryResult</a>	Method for searching for selection objects that meet the conditions.
<a href="#">searchContaining(object)</a>	<a href="#">GeoQueryResult</a>	Method that creates a new selection from objects containing the specified object. Note that many of the geo objects must be added to the map for correct calculations.
<a href="#">searchInside(object)</a>	<a href="#">GeoQueryResult</a>	Method that creates a new selection from objects that are completely included in the specified object. Note that many of the geo objects must be added to the map for correct calculations.
<a href="#">searchIntersect(object[, options])</a>	<a href="#">GeoQueryResult</a>	Method that creates a new selection from objects intersecting the specified object. Note that many of the geo objects must be added to the map for correct calculations.

Name	Returns	Description
<code>setOptions(key[, value])</code>	<code>GeoQueryResult</code>	Method for setting option values for all the selection items.
<code>setProperty(path, value)</code>	<code>GeoQueryResult</code>	Method for setting the value of the "properties" field for all the selection items.
<code>slice(begin[, end])</code>	<code>GeoQueryResult</code>	Method that returns a slice of the selection.
<code>sort(comparator)</code>		Method for sorting selection objects. It does not change the original selection; it creates a new one containing sorted objects.
<code>sortByDistance(object)</code>	<code>GeoQueryResult</code>	Method for getting a selection containing objects sorted by their distance from the specified object. Note that many of the geo objects must be added to the map for correct calculations. It does not change the original selection.
<code>then([onFulfill[, onReject, context])</code>	<code>GeoQueryResult</code>	Subscription to the promise.
<code>unsetOptions(keys)</code>	<code>GeoQueryResult</code>	Method for nullifying option values for all the selection items.
<code>unsetProperties(path)</code>	<code>GeoQueryResult</code>	Method for nullifying the value of the "properties" field for all the selection items.

## Methods details

### add

```
{GeoQueryResult} add(source)
```

Adds objects from the source to objects in the result. It does not change the source object; it creates a new object containing a combined set of geo objects.

**Returns** a new object with the combined set of geo objects.

**Parameters:**

Parameter	Default value	Description
<code>source</code> *	—	<p>Type: Object</p> <p>Source of geo objects:</p> <ul style="list-style-type: none"> <li>IGeoObject - Object that implements the corresponding interface.</li> <li>IGeoObject[] - Array of objects that implement the corresponding interface.</li> <li>ICollection - Collection of objects that implement the IGeoObject interface.</li> <li>ICollection[] - Array of object collections that implement the IGeoObject interface.</li> <li>vow.Promise - A promise object that passes the data source for geoQuery to the handler function. The handler function can also be passed an object with the geoObjects field containing the data source for geoQuery.</li> <li>GeoQueryResult - Object of the GeoQueryResult class.</li> <li>String Object - String or object with a JSON description of objects.</li> <li>Object[] - Array of JSON descriptions of geometries. An array item is an object with the "type" and "coordinates" fields. When describing a circle, the "radius" field is also mandatory. When describing a polygon, the optional "fillRule" field may also be specified.</li> </ul> <p>JSON object descriptions are formed using the following approach (see the example below). An object may be an entity or a collection of entities. A collection of entities is made up of an object with the following fields:</p> <ul style="list-style-type: none"> <li>type - Type of object. The value of the field must be "FeatureCollection".</li> <li>features - Array of child entities in the collection. Child objects may be entities or nested collections of entities.</li> </ul> <p>An entity is made up of an object with the following fields:</p> <ul style="list-style-type: none"> <li>type - Type of object. The value of the field must be</li> </ul>

\* Mandatory parameter/option.

### Examples:

#### 1.

```
// Adding a single geo object to GeoQueryResult.
var placemark = new ymaps.Placemark([34, 56]);
myGeoQueryResult.add(placemark);
```

#### 2.

```
// Adding an array of geo objects to GeoQueryResult.
var objects = [
  new ymaps.Placemark([34, 56]),
  new ymaps.Rectangle([34, 56], [36, 57])
];
// Note that a different GeoQueryResult object will be obtained in the result,
// while the old one remains unchanged.
var newResult = myGeoQueryResult.add(objects);
```

#### 3.

```
// Adding collections of geo objects to GeoQueryResult.
// Note that a different GeoQueryResult object will be obtained in the result,
// while the old one remains unchanged.
var newResult = myGeoQueryResult.add(myMap.geoObjects).searchIntersect(myBounds);
alert("Number of geo objects in the specified area: " + myGeoQueryResult.getLength());
```

#### 4.

```
// Adding geocoding results to GeoQueryResult.
// Note that a different GeoQueryResult will be obtained in the result,
// while the old one remains unchanged.
var newResult = myGeoQueryResult.add(ymaps.geocode('Siromyatnicheskiy lane'));
newResult.searchInside(myGeoBounds);
// Because the data source is asynchronous, we must wait for the result to be processed.
result.then(function () {
  // The source object still exists.
  alert('Before adding objects, the number of objects was: ' + myGeoQueryResult.getLength());
  // The new object has both the old geo objects and the ones that were added.
  alert('After adding objects, the number of objects is: ' + result.getLength());
});
```

#### 5.

```
// Adding objects from a JSON string.
var result = ymaps.geoQuery(myMap.geoObjects);
var extendedResult = result.add('{"type": "Feature", "geometry": { "type": "Point", "coordinates": [15, 64] }}');
extendedResult.addToMap(myMap);
```

## addEvents

```
{GeoQueryResult} addEvents(events, callback, context)
```

Assigns event handlers to selection items.

Returns self-reference.

### Parameters:

Parameter	Default value	Description
<code>events</code> *	—	Type: String String[]  Event type or array of event types that the subscription ends on.
<code>callback</code> *	—	Type: Function  Handler function.

Parameter	Default value	Description
<code>context</code> *	—	Type: Object  Context for the handler function.

\* Mandatory parameter/option.

#### Example:

```
ymaps.geoQuery(map.geoObjects).search('geometry.type="Circle").addEvents('click', function () {
    alert('You clicked a circle!');
});
```

### addTo

```
{GeoQueryResult} addTo(collection)
```

Method for adding objects to a collection of geo objects.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>collection</code> *	—	Type: <a href="#">ICollection</a>  The collection that objects will be added to.

\* Mandatory parameter/option.

#### Example:

```
// Showing objects in the northern hemisphere on the map.
var result1 = ymaps.geoQuery(placemarks).search('lat > 0').addTo(myMap.geoObjects);
```

### addToMap

```
{GeoQueryResult} addToMap(map)
```

Method for adding objects to the map.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>map</code> *	—	Type: <a href="#">Map</a>  The map that objects will be added to.

\* Mandatory parameter/option.

#### Example:

```
// Showing objects in the northern hemisphere on the map.
var result1 = ymaps.geoQuery(placemarks).search('lat > 0').addToMap(myMap);
```

**applyBoundsToMap**

```
{GeoQueryResult} applyBoundsToMap(map[, options])
```

Method for setting the visible area of the map so that all the objects in the selection are visible.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>map</code> *	—	Type: <a href="#">Map</a> Map.
<code>options</code>	—	Type: Object Options.
<code>options.checkZoomRange</code>	false	Type: Boolean Checks whether the specified zoom level can be set. If the value of this option is "true", the method is called asynchronously. A request is sent to the server, which returns the range of acceptable zoom values for the given center. After this, the specified center and appropriate zoom are applied.
<code>options.duration</code>	0	Type: Number Animation duration, in milliseconds.
<code>options.preciseZoom</code>	false	Type: Boolean The ability to use fractional zoom levels.
<code>options.timingFunction</code>	'linear'	Type: String Timing function. The same as the value of the CSS property <code>transition-timing-function</code> . Full list of values: <a href="http://www.w3.org/TR/css3-transitions/#transition-timing-function_tag">http://www.w3.org/TR/css3-transitions/#transition-timing-function_tag</a>
<code>options.useMapMargin</code>	true	Type: Boolean Whether to account for map margins <a href="#">map.margin.Manager</a> .

Parameter	Default value	Description
<a href="#">options.zoomMargin</a>	0	Type: Number Number[]  Offset from the borders of the visible area of the map. If a single number is set, it is applied to each side. If two numbers are set, they are the horizontal and vertical margins, respectively. If an array of four numbers is set, they are the top, right, bottom, and left margins. When the "useMapMargin" option is enabled, the "zoomMargin" value is combined with the values that were calculated in the margins manager <a href="#">map.margin.Manager</a> .

\* Mandatory parameter/option.

### Examples:

#### 1.

```
// Making a geocoding request and setting the visible area of the map
// so that all the results will be visible on it immediately.
var result = ymaps.geoQuery(ymaps.geocode('Lenin street')).applyBoundsToMap(myMap);
// Note that the request to the server is asynchronous and we need to wait for results.
result.then(function () {
  alert("Got results and adjusted the visible area of the map.");
}, function () {
  alert("An error occurred.");
});
```

#### 2.

```
// For synchronous requests, the new map area is set immediately.
var result = ymaps.geoQuery(objects).applyBoundsToMap(myMap);
alert('The visible area of the map changed.');
```

### clusterize

```
{Clusterer} clusterize([options])
```

This method creates a clusterer and adds objects from the selection to it. If the selection data are not ready yet, they will be added to the clusterer immediately after processing, and the returned clusterer will be empty at first. Only objects with the "Point" geometry will be added to the clusterer.

Returns clusterer.

### Parameters:

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object  Options passed to the object constructor <a href="#">Clusterer</a> .

### Example:

```
// Selecting only cafes and adding them to the clusterer.
var clusterer = ymaps.geoQuery(objects).search('properties.type="Cafe").clusterize();
myMap.geoObjects.add(clusterer);
```

## each

```
{GeoQueryResult} each(callback, context)
```

Returns self-reference.

### Parameters:

Parameter	Default value	Description
<code>callback</code> *	—	Type: Function  Handler function. Called for each of the selection items and gets the item as input.
<code>context</code> *	—	Type: Object  Context for the handler function.

\* Mandatory parameter/option.

### Example:

```
// Hiding red placemarks in the visible area of the map.
ymaps.geoQuery(placemarks).searchIntersect(myMap).each(function(pm) {
  if (pm.options.get('preset') == 'islands#redIcon') {
    myMap.geoObjects.remove(pm);
  }
});
```

## get

```
{IGeoObject} get(index)
```

Returns a selection item by index.

### Parameters:

Parameter	Default value	Description
<code>index</code> *	—	Type: Number  Index of items.

\* Mandatory parameter/option.

### Examples:

#### 1.

```
// Example with synchronous processing.
var result = ymaps.geoQuery(placemarks).sort('lat');
// The southernmost object.
var southObject = result.get(0);
// The northernmost object.
northObject = result.get(result.getLength() - 1);
```

#### 2.

```
// Example with asynchronous processing.
var result = ymaps.geoQuery(ymaps.geocode('Moscow cafe')).sort('lat');
// We have to wait for the result to be ready before processing further.
result.then(function () {
  // Southernmost object.
  var southObject = result.get(0);
  // Northernmost object.
  var northObject = result.get(result.getLength() - 1);
});
```

## getBounds

```
{Number[][]|null} getBounds()
```

Returns geographical coordinates of the area that includes all the result objects.

### Examples:

#### 1.

```
// Example of adding the objects synchronously.
// Setting the map center and zoom so that all the objects added to the map are displayed.
var myResult = ymaps.geoQuery(myMap.geoObjects);
myMap.setBounds(myResult.getBounds(), { checkZoomRange: true });
```

#### 2.

```
// Example of adding the objects asynchronously.
// Waiting for the data and only then call the getBounds method.
var myResult = ymaps.geoQuery(ymaps.geocode(' Zveronozhka river')).then(function () {
  myMap.setBounds(myResult.getBounds(), { checkZoomRange: true });
});
```

## getCenter

```
{Number[]} getCenter(map)
```

Method that returns the center of the area that covers all the result objects, in geographical coordinates.

Returns coordinates of the center of the area, in geographical coordinates.

### Parameters:

Parameter	Default value	Description
<code>map</code> *	—	Type: <a href="#">Map</a> The map to make calculations for.

\* Mandatory parameter/option.

### Example:

```
// Moving the map center to the center of the area that covers the objects.
myMap.setCenter(ymaps.geoQuery(objects).getCenter());
```

## getCentralObject

```
{IGeoObject|null} getCentralObject(map)
```

Method for getting the object that is closest to the center of the visible area of the map.

Returns reference to a geo object, or null if the selection is empty.

### Parameters:

Parameter	Default value	Description
<code>map</code> *	—	Type: <a href="#">Map</a> Map.

\* Mandatory parameter/option.

### Examples:

## 1.

```
// Example of working with synchronous operations.
// Opening a balloon at the object closest to the center of the visible area of the map.
ymaps.geoQuery(objects).getCentralObject(myMap).balloon.open();
```

## 2.

```
// Usage example with asynchronous operations.
var result = ymaps.geoQuery(ymaps.geocode('Ulitsa Stroiteley'));
// We just sent the geocoding request so we need to wait for
// the result.
result.then(function () {
    result.getCentralObject(myMap).balloon.open();
});
```

**getClosestTo**

```
{IGeoObject|null} getClosestTo(object)
```

Method that returns the selection object that is closest to the one specified. If an object that is already in the selection is given as input, it returns a different object from the selection that is closest to the one specified. Note that many of the geo objects must be added to the map for correct calculations.

**Returns** the selection object closest to the one specified, or null if the object cannot be found.

**Parameters:**

Parameter	Default value	Description
<code>object</code> *	—	<p>Type: Object</p> <p>The object that the search will be relative to. Accepts the following values:</p> <ul style="list-style-type: none"> <li>IGeoObject - Object that implements the <a href="#">IGeoObject</a> interface.</li> <li>IGeometry - Object that implements the <a href="#">IGeometry</a> interface.</li> <li>Map - The map. In this case, the reference object is the rectangular border of the map.</li> <li>Number[] - Coordinates of a point.</li> <li>Object - JSON description of a geometry. Contains the "type" and "coordinates" fields. When describing a circle, the "radius" field is also mandatory. When describing a polygon, the optional "fillRule" field may also be specified.</li> </ul>

\* Mandatory parameter/option.

**Examples:**

## 1.

```
// Examples of using the method with various input data.
var result = ymaps.geoQuery(objects).addToMap(myMap);
```

```
// 1. IGeoObject.
// A search can be made relative to an outside object.
var polyline = new ymaps.Polyline([[35, 65], [35, 66], [34, 62], [34, 63]]);
myMap.geoObjects.add(polyline);
var closestObject = result.getClosestTo(polyline);
// The search can be based on an object from the same selection.
var closestToFirst = result.getClosestTo(result.get(0));

// 2. IGeometry.
var closestToGeometry = result.getClosestTo(placemark.geometry);

// 3. Map.
// Finding the object closest to the border of the visible area of the map.
var edgeObject = result.getClosestTo(myMap);

// 4. Object nearest a point.
var closestObject = result.getClosestTo([34, 53]);
```

**2.**

```
// Example with asynchronous operations.
var result = ymaps.geoQuery(ymaps.geocode('Paris')).addToMap(myMap);
// Waiting for the server response and getting the object nearest the point.
result.then(function () {
  var closestObject = result.getClosestTo([34, 65]);
  // If the response is empty, the nearest object will not be found.
  if (closestObject) {
    closestObject.balloon.open();
  }
});
```

**getExtreme**

```
{Number} getExtreme(key)
```

Returns the maximum and minimum coordinate values among the coordinates of objects in the selection.

**Parameters:**

Parameter	Default value	Description
<i>key</i> *	—	Type: String  Key for receiving data. Accepts the following values: <ul style="list-style-type: none"> <li>• top</li> <li>• right</li> <li>• bottom</li> <li>• left</li> </ul>

\* Mandatory parameter/option.

**Examples:****1.**

```
// Example with synchronous operations.
alert('Northernmost coordinate: ', ymaps.geoQuery(myMap.geoObjects).getExtreme('top'));
```

**2.**

```
// Example with asynchronous operations.
var result = ymaps.geoQuery(ymaps.geocode('Novgorod'));
// Waiting for the server response and getting the northernmost coordinate.
result.then(function () {
  alert('Northernmost coordinate in the response: ' + result.getExtreme('top'));
});
```

**getExtremeObject**

```
{IGeoObject} getExtremeObject(key)
```

Method that returns the object with the minimum or maximum coordinates among the coordinates of objects in the selection.

**Returns** object with the required coordinate.

**Parameters:**

Parameter	Default value	Description
<code>key *</code>	—	Type: String  Key for searching for an object. Accepts the following values: <ul style="list-style-type: none"> <li>• top</li> <li>• right</li> <li>• bottom</li> <li>• left</li> </ul>

\* Mandatory parameter/option.

**Examples:**

1.

```
// Example with synchronous operations.
// Opening a balloon on the northernmost object.
var topObject = ymaps.geoQuery(myMap.geoObjects).getExtremeObject('top');
topObject.balloon.open();
```

2.

```
// Example with asynchronous operations.
var result = ymaps.geoQuery(ymaps.geocode('Laptev Sea'));
// Waiting for the server response in order to work with data.
result.then(function () {
  var topObject = result.getExtremeObject('top');
  // If the response is empty, the object will not be found.
  if (topObject) {
    topObject.balloon.open();
  }
});
```

## getGlobalPixelBounds

```
{Number[][]|null} getGlobalPixelBounds(map)
```

Method that returns global pixel coordinates of the area for the current map zoom.

**Returns** global pixel coordinates of the area that includes all objects in the result.

**Parameters:**

Parameter	Default value	Description
<code>map *</code>	—	Type: <a href="#">Map</a>  The map that calculations are being made for.

\* Mandatory parameter/option.

**Example:**

```
var result = ymaps.geoQuery(placemarks).search('properties.type="shop"').getGlobalPixelBounds(myMap);
if (Math.abs(result[0][0] - result[1][0]) > myMap.container.getSize()[0]) {
  alert('Objects are too wide to fit on the map!');
}
```

## getGlobalPixelCenter

```
{Number[]} getGlobalPixelCenter(map)
```

**Returns** coordinates of the center of the area, that covers all the result objects, in global pixel coordinates for the current map zoom.

### Parameters:

Parameter	Default value	Description
<code>map</code> *	—	Type: <a href="#">Map</a>  The map to make calculations for.

\* Mandatory parameter/option.

### Example:

```
// Calculating the tile number that contains the center of the area that spans the result.
var globalPixelCenter = ymaps.geoQuery(objects).getGlobalPixelCenter(myMap);
var tileNumber = [
  Math.floor(globalPixelCenter[0] / 256),
  Math.floor(globalPixelCenter[1] / 256)
];
alert('Number of the center tile: ' + tileNumber[0] + ' ' + tileNumber[1]);
```

## getIterator

```
{Iterator} getIterator()
```

**Returns** iterator for objects in the result.

### Examples:

#### 1.

```
// Using the iterator with synchronous operations.
// Searching for elements that match the click coordinates.
myMap.events.add('click', function (event) {
  var iterator = ymaps.geoQuery(myMap.geoObjects)
    .searchContaining(event.getCoordinates())
    .getIterator();
  var obj;
  while ((obj = iterator.getNext()) != iterator.STOP_ITERATION) {
    // Performing the necessary actions on a geo object.
  }
});
```

#### 2.

```
// Using an iterator with asynchronous operations.
// Creating a result from the geocoder query.
var result = ymaps.geoQuery(ymaps.geocode("Starokolpakskiy street")).search("lat > 20");
// Note that we only sent the request to the server, and the response will come later.
// Because the request is asynchronous, we need to wait for it to be ready before getting the result.
result.then(function () {
  var iterator = result.getIterator();
  var obj;
  while ((obj = iterator.getNext()) != iterator.STOP_ITERATION) {
    // Performing the necessary actions on the geo object.
  }
});
```

## getLength

```
{Number} getLength()
```

**Returns** number of items in the result.

### Examples:

## 1.

```
var result = ymaps.geoQuery(myMap.geoObject).searchIntersect(myPolygon);
alert('The number of geo objects that intersect with the polygon: ' + result.getLength());
```

## 2.

```
var result = ymaps.geoQuery(ymaps.geocode('Ivanovo')).searchInside(myMap);
// Because we sent the request to the server, we cannot immediately count
// the number of elements in the result. We have to wait for the response.
result.then(function () {
  alert('Number of objects in the visible area of the map: ' + result.getLength());
});
```

**getMaxZoom**

```
{Number} getMaxZoom(map[, options])
```

Method that calculates the maximum zoom level at which all the objects fall within the visible area of the map.

**Returns** maximum map zoom level.

**Parameters:**

Parameter	Default value	Description
<code>map</code> *	—	Type: <a href="#">Map</a>  The map that the calculation is being made for.
<code>options</code>	—	Type: Object  Options.
<code>options.useMapMargin</code>	true	Type: Boolean  Whether to account for map margins <a href="#">map.margin.Manager</a> .

\* Mandatory parameter/option.

**Example:**

```
// Calculating the maximum zoom level
// at which all the selection objects are visible
// and setting a restriction for the map.
var maxZoom = ymaps.geoQuery(objects).getMaxZoom();
myMap.options.set('maxZoom', maxZoom);
```

**getParent**

```
{GeoQueryResult|null} getParent()
```

**Returns** reference to the parent selection, if the current selection was created as the result of changing another `GeoQueryResult` object.

**Example:**

```
ymaps.geoQuery(objectsArray)
  // Selecting only objects of the type "cafe" and defining their styles.
  .search('properties.type == "cafe"')
  .setOptions('preset', 'islands#yellowDotIcon')
  // Then, without interrupting the chain of calls,
  // getting back to the original selection and defining styles for another group of objects.
  .getParent()
  .search('properties.type == "shop"')
  .setOptions('preset', 'islands#greenDotIcon');
```

## indexOf

```
{Number} indexOf(item)
```

**Returns** index of the item in the selection. If the item was not found, it returns -1.

### Parameters:

Parameter	Default value	Description
<i>item</i> *	—	Type: <a href="#">IGeoObject</a> The required object.

\* Mandatory parameter/option.

### Example:

```
// Sorting the selection by the "name" field.
var result = ymaps.geoQuery(polygons).sort('properties.name');
alert('New position of the first item: ' + result.indexOf(polygons[0]));
```

## intersect

```
{GeoQueryResult} intersect(result)
```

This method creates a new selection containing common items for two different selections.

**Returns** the new selection that contains the intersection result.

### Parameters:

Parameter	Default value	Description
<i>result</i> *	—	Type: <a href="#">GeoQueryResult</a> The selection to intersect the first one with.

\* Mandatory parameter/option.

### Examples:

#### 1.

```
// Example of intersection with synchronous operations.
var result = ymaps.geoQuery(placemarks);
var greenObjects = result.search('properties.color="green"');
var roundObjects = result.search('properties.shape="round"');
var greenRoundObjects = greenObjects.intersect(roundObjects);
alert('Number of round green objects: ' + greenRoundObjects.getLength());
```

#### 2.

```
// Example of intersection with asynchronous operations.
var result = ymaps.geoQuery(ymaps.geocode('Ivanovka'));
var filteredByLat = result.search('lat > 56');
var filteredByLong = result.search('long > 36');
var intersectedResult = filteredByLat.intersect(filteredByLong);
// Because the original request is asynchronous, we must wait for data to be ready.
intersectedResult.then(function () {
  alert('Number of objects with the name "Ivanovka" +
    'with coordinates higher than [56, 36]: ' +
    intersectedResult.getLength());
});
```

## isReady

```
{Boolean} isReady()
```

**Returns** whether the selection results are ready or are still being processed.

**Example:**

```
var result = ymaps.geoQuery(ymaps.geocode('Ivanovo'));
if (!result.isReady()) {
  result.then(function () {
    // Processing data.
  });
} else {
  // Processing data.
}
```

## map

```
{GeoQueryResult} map(callback[, context])
```

Method that calls the "callback" method for all the selection items and forms a new selection based on the results received.

**Returns** the new selection.

**Parameters:**

Parameter	Default value	Description
<code>callback</code> *	—	Type: Function  Handler function. Takes a selection item as input. Returns an instance of <a href="#">IGeoObject</a> .
<code>context</code>	—	Type: Object  Context for the handler function.

\* Mandatory parameter/option.

**Example:**

```
// Adding only circle objects to the map.
var circlesResult = ymaps.geoQuery(objects).search('geometry.type="Circle"').addToMap(myMap);
// We'll also add placemarks that indicate the circle centers.
var centers = circlesResult.map(function (object) {
  return new ymaps.Placemark(object.geometry.getCenter());
}).addToMap(myMap);
```

## remove

```
{GeoQueryResult} remove(objects)
```

Deletes objects from the result. It does not change the source object; it creates a new object containing the resulting set of geo objects.

**Returns** new object with the resulting set of geo objects.

**Parameters:**

Parameter	Default value	Description
<code>objects</code> *	—	<p>Type: Object</p> <p>Objects can be presented in various forms:</p> <ul style="list-style-type: none"> <li>• <code>IGeoObject</code> - Object that implements the corresponding interface.</li> <li>• <code>IGeoObject[]</code> - Array of objects that implement the corresponding interface.</li> <li>• <code>ICollection</code> - Collection of objects that implement the <code>IGeoObject</code> interface.</li> <li>• <code>ICollection[]</code> - Array of object collections that implement the <code>IGeoObject</code> interface.</li> <li>• <code>GeoQueryResult</code> - Object of the <code>GeoQueryResult</code> class. Note that with asynchronous operations, the result has to be prepared before it can be deleted correctly.</li> <li>• <code>vow.Promise</code> - Object of the <code>vow.Deferred</code> class. Must be resolved by an array of geo objects or an object with the "geoObjects" field.</li> </ul>

\* Mandatory parameter/option.

### Examples:

#### 1.

```
var objects = [
  new ymaps.Placemark([34, 56]),
  new ymaps.Rectangle([34, 56], [36, 57])
];
var result = ymaps.geoQuery(objects);
// Note that a different GeoQueryResult object will be obtained in the result,
// while the old one remains unchanged.
var newResult = result.remove(objects[1]);
```

#### 2.

```
// Deleting an array of geo objects from GeoQueryResult.
var objects = [
  new ymaps.Placemark([34, 56]),
  new ymaps.Rectangle([34, 56], [36, 57]),
  new ymaps.Placemark([35, 64])
];
var result = ymaps.geoQuery(objects);
// Note that a different GeoQueryResult object will be obtained in the result,
// while the old one remains unchanged.
var newResult = result.remove([objects[1], objects[2]]);
```

#### 3.

```
// Deleting collections of geo objects from GeoQueryResult.
// Adding objects to the map that aren't on it yet.
myGeoQueryResult.remove(myMap.geoObjects).addToMap(myMap);
```

## 4.

```
// Deleting data of a different GeoQueryResult from GeoQueryResult.
var result1 = ymaps.geoQuery(placemarks).search('properties.color="green"');
var result2 = ymaps.geoQuery(placemarks).search('properties.shape="circle"');
var result3 = result1.remove(result2);
alert('Number of green non-round shapes: ' + result3.getLength());
```

**removeEvents**

```
{ } removeEvents(events, callback, context)
```

Deletes the subscription to events from an object. Note that in order to unsubscribe correctly, the arguments passed must be exactly the same as the ones for subscribing via the "addEvents" method.

**Parameters:**

Parameter	Default value	Description
<code>events *</code>	—	Type: String String[]  Event type or array of types that a subscription was made to.
<code>callback *</code>	—	Type: Function  Handler function that was specified when subscribing.
<code>context *</code>	—	Type: Object  Context that was specified when subscribing.

\* Mandatory parameter/option.

**Example:**

```
var callback = function () {
  alert('You clicked a circle!');
};
ymaps.geoQuery(map.geoObjects).search('geometry.type="Circle"').addEvents('click', callback);
// ...
ymaps.geoQuery(map.geoObjects).search('geometry.type="Circle"').removeEvents('click', callback);
```

**removeFrom**

```
{GeoQueryResult} removeFrom(collection)
```

Method for deleting objects from a collection.

Returns self-reference.

**Parameters:**

Parameter	Default value	Description
<code>collection *</code>	—	Type: <a href="#">ICollection</a>  The collection to delete objects from.

\* Mandatory parameter/option.

**Example:**

```
// Showing all objects on the map.
var result1 = ymaps.geoQuery(placemarks).addTo(myMap.geoObjects);
// Then hiding objects in the northern hemisphere.
var result2 = result1.search('lat > 0').removeFrom(myMap.geoObjects);
```

**removeFromMap**

```
{GeoQueryResult} removeFromMap(map)
```

Method for deleting objects from the map.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>map</code> *	—	Type: <a href="#">Map</a>  The map to delete objects from.

\* Mandatory parameter/option.

**Example:**

```
// Showing all objects on the map.
var result1 = ymaps.geoQuery(placemarks).addToMap(myMap);
// Then hiding objects in the northern hemisphere.
var result2 = result1.search('lat > 0').removeFromMap(myMap);
```

**reverse**

```
{GeoQueryResult} reverse()
```

Rearranges the selection items in reverse order and returns the new selection.

**Returns** the new selection with items in reverse order.

**Examples:****1.**

```
// Usage with synchronous requests.
var result = ymaps.geoQuery(myMap.geoObjects).sort('x'),
    invertedResult = result.reverse();
```

**2.**

```
// Usage with asynchronous requests.
var result = ymaps.geoQuery(ymaps.geocode('Friendly Bees village')).sort('x');
var invertedResult = result.reverse();
invertedResult.then(function () {
    // Got the result. Ready to use.
});
```

**search**

```
{GeoQueryResult} search(condition)
```

Method for searching for selection objects that meet the conditions.

**Returns** a new selection containing search results.

**Parameters:**

Parameter	Default value	Description
<code>condition</code> *	—	<p>Type: String Function</p> <p>String template for search or a filter function. A string template has the structure "<code>&lt;fieldname&gt; &lt;condition&gt; &lt;expression&gt;</code>". Acceptable values for <code>&lt;fieldname&gt;</code>:</p> <ul style="list-style-type: none"> <li>'lat' - Latitude. Supported only for point objects.</li> <li>'lng', 'long' - Longitude. Supported only for point objects.</li> <li>'x' - Global pixel coordinates on the X axis. Supported only for point objects.</li> <li>'y' - Global pixel coordinates on the Y axis. Supported only for point objects.</li> <li>'geometry.type' - Type of geometry. Result of executing the <code>geometry.getType()</code> method.</li> <li>'geometry.coordinates.&lt;index&gt;' - Coordinates. Result of executing the <code>geometry.getCoordinates()</code> method. To get access to a coordinate, specify its indexes after a dot - 'geometry.coordinates.0.1'.</li> <li>'properties.&lt;path&gt;' - Value of the data field.</li> <li>'options.&lt;key&gt;' - Value of options.</li> </ul> <p>Acceptable values for <code>&lt;condition&gt;</code>:</p> <ul style="list-style-type: none"> <li>'&gt;'</li> <li>'&gt;='</li> <li>'==', '!='</li> <li>'!='</li> <li>'&lt;='</li> <li>'&lt;'</li> <li>'rlike', 'regexp' - Matches the <code>&lt;fieldname&gt;</code> value to a regular expression.</li> </ul> <p>Acceptable values for <code>&lt;expression&gt;</code>:</p> <ul style="list-style-type: none"> <li>Number</li> <li>String (must be put in quotation marks)</li> <li>true</li> <li>false</li> <li>null</li> <li>undefined</li> </ul> <p>The filter function takes a selection object as input and returns true/false. True means the object is in the</p>

\* Mandatory parameter/option.

### Examples:

#### 1.

```
// Example of searching for objects with synchronous operations.
var result = ymaps.geoQuery(myMap.geoObjects);

    // Searching for objects with a specific type of geometry. Note that
// the field value shown in quotes is a string.
result.search('geometry.type = "Circle"')
// Searching by a coordinate.
    .search('geometry.coordinates.0 > 100')
// Searching by latitude.
    .search('lat < 0')
// Searching by pixel coordinates.
    .search('x >= 100')
// Searching by the field value in "properties".
    .search('properties.name != null').search('properties.name rlike "(.) \\\"")')
    .search('properties.author.name = "Stepan"')
// Searching by option value.
    .search('options.visible = true');
```

#### 2.

```
// Using the method with asynchronous operations.
var result = ymaps.geoQuery(ymaps.geocode('Moose'));
// Waiting for the result to be ready before working with it.
result.then(function () {
    result.search('properties.description regexp "*village*")
        .addToMap(myMap)
        .applyBoundsToMap(myMap);
});
```

### searchContaining

```
{GeoQueryResult} searchContaining(object)
```

Method that creates a new selection from objects containing the specified object. Note that many of the geo objects must be added to the map for correct calculations.

**Returns** a new selection containing the desired objects.

**Parameters:**

Parameter	Default value	Description
<code>object *</code>	—	<p>Type: Object</p> <p>The object that the search will be relative to. Accepts the following values:</p> <ul style="list-style-type: none"> <li>• <code>IGeoObject</code> - Object that implements the <a href="#">IGeoObject</a> interface.</li> <li>• <code>IGeometry</code> - Object that implements the <a href="#">IGeometry</a> interface.</li> <li>• <code>Map</code> - The map. In this case, the reference object is the rectangular border of the map.</li> <li>• <code>Number[]</code> - Coordinates of a point. If one parameter is passed, it is interpreted as a point. If two arguments are passed, it is interpreted as a circle.</li> <li>• <code>Number[][]</code> - Coordinates of a rectangular area.</li> <li>• <code>Object</code> - JSON descriptions of a geometry.</li> </ul>

\* Mandatory parameter/option.

## Examples:

### 1.

```
// Examples of using the method with various input data.
var result = ymaps.geoQuery(objects).addToMap(myMap);

// 1. IGeoObject.
var polygon = new ymaps.Polygon([[35, 65], [35, 66], [34, 62], [34, 63], [35, 65]]);
myMap.geoObjects.add(polygon);
var objectsContainingPolygon = result.searchContaining(polygon);

// 2. IGeometry.
var objectsContainingGeometry = result.searchContaining(polygon.geometry);

// 3. Map.
var objectsContainingMapBounds = result.searchContaining(myMap);
```

### 2.

```
// Example of usage with asynchronous operations.
var result = ymaps.geoQuery(ymaps.geocode('cafe'))
    .addToMap(myMap);
// Waiting for the server response before processing results.
result.then(function () {
    var areas = result.map(function (object) {
        return new ymaps.Circle(object.geometry.getCoordinates(), 100);
    })
    .setOptions('visible', false)
    .addToMap(myMap);
    myMap.events.add('click', function (event) {
        if (areas.searchContaining(event.getCoordinates()).getLength() {
            alert('You clicked near a cafe.');
```

## searchInside

```
{GeoQueryResult} searchInside(object)
```

Method that creates a new selection from objects that are completely included in the specified object. Note that many of the geo objects must be added to the map for correct calculations.

**Returns** a new selection containing the desired objects.

**Parameters:**

Parameter	Default value	Description
<code>object *</code>	—	<p>Type: Object</p> <p>The object that the search will be relative to. Accepts the following values:</p> <ul style="list-style-type: none"> <li>• IGeoObject - Object that implements the <a href="#">IGeoObject</a> interface.</li> <li>• IGeometry - Object that implements the <a href="#">IGeometry</a> interface.</li> <li>• Map - The map. In this case, the reference object is the rectangular border of the map.</li> </ul>

\* Mandatory parameter/option.

**Examples:**

1.

```
// Examples of using the method with various input data.
var result = ymaps.geoQuery(objects).addToMap(myMap);

// 1. IGeoObject.
var polygon = new ymaps.Polygon([[35, 65], [35, 66], [34, 62], [34, 63], [35, 65]]);
myMap.geoObjects.add(polygon);
var objectsInsidePolygon = result.searchInside(polygon);

// 2. IGeometry.
var objectsInsideGeometry = result.searchInside(polygon.geometry);

// 3. Map.
// We'll find objects that fit entirely inside the visible area of the map.
var visibleObject = result.searchInside(myMap);
```

2.

```
// Example of usage with asynchronous operations.
var result = ymaps.geoQuery(ymaps.geocode('Ivanovo'))
  .setOptions('visible', false)
  .addToMap(myMap);
// Waiting for the server response and then showing objects that fit inside the visible area of the map.
result.then(function () {
  result.searchInside(myMap).setOptions('visible', true);
});
```

**searchIntersect**

```
{GeoQueryResult} searchIntersect(object[, options])
```

Method that creates a new selection from objects intersecting the specified object. Note that many of the geo objects must be added to the map for correct calculations.

**Returns** a new selection containing the desired objects.

**Parameters:**

Parameter	Default value	Description
<code>object *</code>	—	Type: Object  The object that the search will be relative to. Accepts the following values: <ul style="list-style-type: none"> <li>• <code>IGeoObject</code> - Object that implements the <a href="#">IGeoObject</a> interface.</li> <li>• <code>IGeometry</code> - Object that implements the <a href="#">IGeometry</a> interface.</li> <li>• <code>Map</code> - The map. In this case, the reference object is the rectangular border of the map.</li> </ul>
<code>options</code>	—	Type: Object  Options.
<code>options.considerOccurance</code>	<code>true</code>	Type: Object  Flag indicating whether we consider it an intersection when one figure is completely contained inside another.

\* Mandatory parameter/option.

## Examples:

### 1.

```
// Examples of using the method with various input data.
var result = ymaps.geoQuery(objects).addToMap(myMap);

// 1. IGeoObject.
var polygon = new ymaps.Polygon([[35, 65], [35, 66], [34, 62], [34, 63], [35, 65]]);
myMap.geoObjects.add(polygon);
var objectsIntersectPolygon = result.searchIntersect(polygon);

// 2. IGeometry.
var objectsIntersectGeometry = result.searchIntersect(polygon.geometry);

// 3. Map.
// We will only search for objects that directly intersect
// the map border. In other words, objects that are entirely inside the visible area of the map
// will not be in the final selection.
var objectsIntersectMapBounds = result.searchIntersect(myMap, {considerOccurance: false});
```

### 2.

```
// Example of usage with asynchronous operations.
var result = ymaps.geoQuery(ymaps.geocode('cafe'))
  .setOptions('visible', false)
  .addToMap(myMap);

// Waiting for the server response and then showing objects that fall within the map viewport.
result.then(function () {
  result.searchIntersect(myMap).setOptions('visible', true);
});
```

## setOptions

```
{GeoQueryResult} setOptions(key[, value])
```

Method for setting option values for all the selection items.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>key *</code>	—	Type: String Object  Option name or hash with options and their values.
<code>value</code>	—	Type: Object  Option value.

\* Mandatory parameter/option.

**Example:**

```
var result = ymaps.geoQuery(placemarks);
// Making elements visible if they fall within a rectangular area.
result.searchIntersect(myBounds).setOptions('visible', true);

// Setting options using a hash.
result.setOptions({zIndex: 10, fillColor: '#ff0005'});
```

**setProperties**

```
{GeoQueryResult} setProperties(path, value)
```

Method for setting the value of the "properties" field for all the selection items.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>path *</code>	—	Type: String  Name of the field that the value is assigned to. It may contain ".".
<code>value *</code>	—	Type: Object  Field value.

\* Mandatory parameter/option.

**Example:**

```
var result = ymaps.geoQuery(objects);
// Marking elements that fall inside the area.
result.searchIntersect(myBounds1).setProperties('intersectBounds', true);
result.searchIntersect(myBounds2).setProperties('intersectBounds', true);
// ...
result.search('properties.intersectBounds = true').addToMap(myMap);
```

**slice**

```
{GeoQueryResult} slice(begin[, end])
```

Method that returns a slice of the selection.

**Returns** the new selection containing items in the slice.

**Parameters:**

Parameter	Default value	Description
<code>begin</code> *	—	Type: Number  Index of the first item in the slice.
<code>end</code>	—	Type: Number  Index of the selection item that ends the slice. However, the last item in the new slice will be the item with the index end-1.

\* Mandatory parameter/option.

### Examples:

#### 1.

```
// Making a slice with asynchronous processing.
var result = ymaps.geoQuery(map.geoObjects).slice(0, 10);
alert('Number of items in new selection: ' + result.getLength());
```

#### 2.

```
// Getting a slice with asynchronous processing.
var result = ymaps.geoQuery(ymaps.geocode('cafe Moscow')).slice(0, 10);
// The result is not ready immediately after the request.
alert('The selection is still empty. Number of items in selection: ' + result.getLength());
// Waiting for the result to be ready and then we will see how many items will be in the selection.
result.then(function () {
  alert('Response received. Amount of data in the selection: ' + result.getLength());
});
```

### sort

```
{ } sort(comparator)
```

Method for sorting selection objects. It does not change the original selection; it creates a new one containing sorted objects.

### Parameters:

Parameter	Default value	Description
<code>comparator</code> *	—	<p>Type: String Function</p> <p>String with a sorting template or a compare function. The template string can be in the format '<code>&lt;fieldname&gt; [&lt;order&gt;=asc</code>']. Acceptable values for <code>&lt;fieldname&gt;</code>:</p> <ul style="list-style-type: none"> <li>'lat' - Latitude. Supported only for point objects.</li> <li>'lng', 'long' - Longitude. Supported only for point objects.</li> <li>'x' - Global pixel coordinates on the X axis. Supported only for point objects.</li> <li>'y' - Global pixel coordinates on the Y axis. Supported only for point objects.</li> <li>'geometry.type' - Type of geometry. Result of executing the <code>geometry.getType()</code> method.</li> <li>'geometry.coordinates.&lt;index&gt;' - Coordinates. Result of executing the <code>geometry.getCoordinates()</code> method. To get access to a coordinate, specify its indexes after a dot - 'geometry.coordinates.0.1'.</li> <li>'properties.&lt;path&gt;' - Value of the data field.</li> <li>'options.&lt;key&gt;' - Value of options.</li> </ul> <p>Acceptable values for the optional <code>&lt;order&gt;</code>: parameter:</p> <ul style="list-style-type: none"> <li>'asc' - Sort in ascending order.</li> <li>'desc' - Sort in descending order.</li> </ul> <p>The compare function takes two selection items as input. Returned values: If the first object is larger than the second, the function returns the value <code>&gt; 0</code>. If the first object is equal to the second, the function returns <code>0</code>. If the first object is smaller than the second, the function returns <code>&lt; 0</code>.</p>

\* Mandatory parameter/option.

**Example:**

```
// Example with synchronous operations.
var result = ymaps.geoQuery(myMap.geoObjects);
result.sort('lat').sort('x')
  .sort('properties.name desc')
  .sort('options.preset')
  .sort(function (a, b) {
    if (a.properties.get('name') == b.properties.get('name')) {
      return a.geometry.getCoordinates()[0] - b.geometry.getCoordinates()[0];
    } else {
      return (a.properties.get('name') > b.properties.get('name')) ? 1 : -1;
    }
  });
```

**sortByDistance**

```
{GeoQueryResult} sortByDistance(object)
```

Method for getting a selection containing objects sorted by their distance from the specified object. Note that many of the geo objects must be added to the map for correct calculations. It does not change the original selection.

**Returns** the new ordered selection.

**Parameters:**

Parameter	Default value	Description
<a href="#">object</a> *	—	<p>Type: Object</p> <p>The object that the distance will be calculated to. Accepts the following values:</p> <ul style="list-style-type: none"> <li>IGeoObject - Object that implements the <a href="#">IGeoObject</a> interface.</li> <li>IGeometry - Object that implements the <a href="#">IGeometry</a> interface.</li> <li>Map - The map. In this case, the reference object is the rectangular border of the map.</li> <li>Number[] - Coordinates of a point.</li> <li>Number[][] - Coordinates of a rectangular area.</li> <li>Object - JSON description of a geometry. Contains the "type" and "coordinates" fields. When describing a circle, the "radius" field is also mandatory. When describing a polygon, the optional "fillRule" field may also be specified.</li> </ul>

\* Mandatory parameter/option.

**Examples:****1.**

```
// Examples of using the method with various input data.
var result = ymaps.geoQuery(objects).addToMap(myMap);
```

```
// 1. IGeoObject.
var polyline = new ymaps.Polyline([[35, 65], [35, 66], [34, 62], [34, 63]]);
myMap.geoObjects.add(polyline);
var sortedByPolyline = result.sortByDistance(polyline);

// 2. IGeometry.
var sortedByGeometry = result.sortByDistance(placemark.geometry);

// 3. Map.
var sortedByMapBounds = result.sortByDistance(myMap);

// 4. The selection sorted by a point.
var sortedByPoint = result.sortByDistance([34, 53]);
```

## 2.

```
// Example with asynchronous operations.
var result = ymaps.geoQuery(ymaps.geocode('Paris')).addToMap(myMap).sortByDistance([45, 64]);
// Waiting for the result from the server and getting the nearest and furthest
// object relative to the point.
result.then(function () {
  alert('The nearest object has the coordinates ' + result.get(0).geometry.getCoordinates());
  alert('The furthest object has the coordinates ' + result.get(result.getLength() -
1).geometry.getCoordinates());
});
```

## then

```
{GeoQueryResult} then([onFulfill[, onReject, context])
```

Subscription to the promise.

**Returns** self-reference.

### Parameters:

Parameter	Default value	Description
<code>onFulfill</code>	—	Type: Function  Handler function that is called if the promise was fulfilled.
<code>onReject</code>	—	Type: Function  Handler function that is called if the promise was not fulfilled (an error occurred).
<code>context *</code>	—	Type: Object  Context for the handler function.

\* Mandatory parameter/option.

### Example:

```
var result = ymaps.geoQuery(ymaps.geocode('Lena river'));
result.then(function () {
  alert('Number of objects found: ' + result.getLength());
}, function () {
  alert('Error occurred.');
```

## unsetOptions

```
{GeoQueryResult} unsetOptions(keys)
```

Method for nullifying option values for all the selection items.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>keys *</code>	—	Type: String String  Name or array of names of options that should be canceled.

\* Mandatory parameter/option.

**Example:**

```
result.unsetOptions('visible');
```

**unsetProperties**

```
{GeoQueryResult} unsetProperties(path)
```

Method for nullifying the value of the "properties" field for all the selection items.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>path *</code>	—	Type: String  Name of a field to cancel the value for. It may contain ".".

\* Mandatory parameter/option.

**Example:**

```
var result = ymaps.geoQuery(objects);
// Marking items that fall inside the first area, but do not fall inside the second one.
result.searchIntersect(myBounds1).setProperties('intersectBounds', true);
result.searchIntersect(myBounds2).unsetProperties('intersectBounds', true);
// ...
result.search('properties.intersectBounds = true').addToMap(myMap);
```

## geoXml

### geoXml.load

Static function.

Loads an XML file with geographic data and converts it into a [GeoObjectCollection](#). The generated collection can be passed to the specified function for subsequent processing. Supported XML data formats: YMapsML, KML, GPX. For the topmost collection of geo objects from a GPX file, the following presets are available:

- 'gpx#interactive' - Provides outputting information about a point on a route when clicked. Additionally, when this preset is used, the following geo object properties become available in the balloon layout: time, velocity, trackName, trackDescription, pointName, pointDescription, lon, lat, sym. Used by default.
- 'gpx#plain' - Items in GPX collections behave like normal geo objects.

**Returns** Promise object. If the XML file at the specified URL is downloaded successfully, the promise will be resolved and will get an object with the following fields (as parameters):

- geoObjects - Collection of geo objects [GeoObjectCollection](#).

- `mapState` - Description of the map state [IMapState](#) (only for YMapsML).

```
{ vow.Promise } geoXml.load(url)
```

#### Parameters:

Parameter	Default value	Description
<code>url</code> *	—	Type: String  The URL of a file with geographical data.

\* Mandatory parameter/option.

#### Example:

```
// Creating and initializing map...

// Loading and displaying a ymapsml file from the My Maps service
ymaps.geoXml.load('http://maps.yandex.ru/export/usermaps/HNQ5uTUgbyy6L0dW2uReUjSoXb1Ad7jw/')
  .then(function (res) {
    // Adding elements of the ymapsml file to the map
    map.geoObjects.add(res.geoObjects);
    // Setting the map boundaries and type.
    // res.mapState.getBounds() boundaries are applied to the map asynchronously,
    // because we need to get information about available zoom levels for these bounds
    // res.mapState.getType() is applied to the map synchronously.
    if (res.mapState) {
      res.mapState.applyToMap(map).then(function () {
        alert('Boundaries applied to the map ' + res.mapState.getBounds().toString());
      });
    }
    // If information about boundaries isn't provided in repr:View in the YMapsML file,
    // we can apply gml:boundedBy for the top ymaps:GeoObjectCollection element
    else if (res.geoObjects.properties.get('boundedBy')) {
      map.setBounds(res.geoObjects.properties.get('boundedBy'), {
        checkZoomRange: true
      });
    }
  });

// Loading and displaying a KML file
ymaps.geoXml.load('http://api.yandex.ru/maps/doc/jsapi/1.x/examples/kml/demonstration.xml')
  .then(function (res) {
    map.geoObjects.add(res.geoObjects);
  });

// Loading and displaying a GPX file
ymaps.geoXml.load('http://karmatsky.narod2.ru/MskChel2.xml')
  .then(function (res) {
    res.geoObjects.options.set({
      balloonContentBodyLayout: ymaps.templateLayoutFactory.createClass(
        // The balloon will only show the geo object's name property and the speed
        '<b>{{ properties.name }}</b> <b>{{ properties.velocity }}</b>'
      )
    });

    map.geoObjects.add(res.geoObjects);
    // Metadata about boundaries from a GPX file is stored in properties of the res.geoObjects collection.
    // Applying these boundaries to the map.
    if (res.geoObjects.properties.get('boundedBy')) {
      map.setBounds(res.geoObjects.properties.get('boundedBy'), {
        checkZoomRange: true
      });
    }
  });
});
```

## getZoomRange

Static function.

Checks the available range of zoom levels at the specified point for the specified map type.

**Returns** a Promise, which will be resolved and will get an array of two numbers as a parameter - the maximum and minimum zoom at the given point.

```
{ vow.Promise } getZoomRange(mapType, coords, customizable)
```

**Parameters:**

Parameter	Default value	Description
<code>mapType *</code>	—	Type: String  <a href="#">MapType</a>  Map type. Key string from <a href="#">mapType.storage</a> , or an instance of the <a href="#">MapType</a> class.
<code>coords *</code>	—	Type: Number[]  Coordinates of the point to find out the available range of zoom levels for.
<code>customizable *</code>	—	Type: <a href="#">ICustomizable</a> =null  Object that contains the options manager. The object options will be considered when getting the result.

\* Mandatory parameter/option.

**Examples:****1.**

```
// Let's say we want to initialize the map at the maximum zoom.
var myMap;
ymaps.getZoomRange('yandex#map', [55.750516, 37.615924]).then(function (result) {
  myMap = new ymaps.Map('mapContainer', {
    center: [55.750516, 37.615924],
    zoom: result[1]
  });
});
```

**2.**

```
// Initialize the map using the geocoder, centered at Lev Tolstoy street, number 16
// at the maximum zoom possible.
var myMap;
ymaps.geocode("Moscow, Lev Tolstoy, 16").then(function (geoData) {
  var coords = geoData.geoObjects.get(0).geometry.getCoordinates();
  ymaps.getZoomRange('yandex#map', coords).then(function (zoomRange) {
    myMap = new ymaps.Map('mapContainer', {
      center: coords,
      zoom: zoomRange[1]
    });
  });
});
```

## graphics

### graphics.style

#### graphics.style.color

Static object.

Sets the color of a graphic shape in the formats `#RGB`,`#RGBA`,`#RRGGBB`,`#RRGGBBAA`,`rgb(r,b,a)`,`rgba(r,g,b,a)`

**Example:**

```
strokeColor: '#F00';
strokeColor: '#FF0000';
strokeColor: '#FF0000AA';
strokeColor: 'rgba(255,0,0,1)'
```

## graphics.style.stroke

Static object.

For changing line style. The value can be set by using the keys described below, or in array format. It is worth noting that in VML(IE<9) display mode, only keys may be used. Additionally, in the line style, the offset for beginning a dotted line can be set via the "offset" field.

### Fields

#### Example:

```
strokeStyle: 'dot';
strokeStyle: [1,2];
strokeStyle: {
  style: 'dot',
  offset: 10
}
```

### Fields

Name	Type	Description
<a href="#">dash</a>		Dash
<a href="#">dashdot</a>		Long dash - short dash
<a href="#">dot</a>		Dots
<a href="#">longdash</a>		Long dashes
<a href="#">longdashdot</a>		Extra long dash - dot
<a href="#">longdashdotdot</a>		Long dash - dot - dot
<a href="#">shortdash</a>		Short dashes
<a href="#">shortdashdot</a>		Dash - dot
<a href="#">shortdashdotdot</a>		Dash - dot - dot
<a href="#">shortdot</a>		Dots with double spacing
<a href="#">solid</a>		Solid line

### Fields details

#### dash

```
dash
```

Dash

#### dashdot

```
dashdot
```

Long dash - short dash

#### dot

```
dot
```

Dots

**longdash**

```
longdash
```

Long dashes

**longdashdot**

```
longdashdot
```

Extra long dash - dot

**longdashdotdot**

```
longdashdotdot
```

Long dash - dot - dot

**shortdash**

```
shortdash
```

Short dashes

**shortdashdot**

```
shortdashdot
```

Dash - dot

**shortdashdotdot**

```
shortdashdotdot
```

Dash - dot - dot

**shortdot**

```
shortdot
```

Dots with double spacing

**solid**

```
solid
```

Solid line

## Hint

Extends [IHint](#), [Popup](#).

A hint is a popup text that can display any HTML content. There is usually just one hint instance on the map and it is managed via special managers ([maps](#), [geo objects](#), [hotspot layers](#) and so on). Don't create them yourself, unless truly necessary.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
Hint(map[, options])
```

**Parameters:**

Parameter	Default value	Description
<code>map</code> *	—	Type: <a href="#">Map</a> Reference to a map object.
<code>options</code>	—	Type: Object Options.
<code>options.closeTimeout</code>	700	Type: Number Delay before closing (in ms).
<code>options.contentLayout</code>	—	Type: Function String Layout for hint content. (Type: constructor for an object with the <a href="#">ILayout</a> interface).
<code>options.fitPane</code>	true	Type: Boolean Flag that forces the info object to move its position so that it doesn't extend beyond the container boundaries.
<code>options.holdByMouse</code>	true	Type: Boolean Flag that cancels closing a hint that is located under the cursor.
<code>options.interactivityModel</code>	—	Type: String Key for the interactivity model. Available keys and their values are listed in the description of <a href="#">interactivityModel.storage</a> .
<code>options.layout</code>	islands#hint	Type: Function String External layout for the hint. (Type: constructor for an object with the <a href="#">ILayout</a> interface).
<code>options.offset</code>	—	Type: Number[] Additional position offset relative to the anchor point.
<code>options.openTimeout</code>	150	Type: Number Delay before opening (in ms).
<code>options.pane</code>	'outerHint'	Type: String Key of the pane that the hint overlay is placed in.

Parameter	Default value	Description
<a href="#">options.zIndex</a>	—	Type: String The z-index of the hint.

\* Mandatory parameter/option.

#### Example:

```
// Creating an independent hint instance and showing it with a custom layout in the center of the map by setting the
// geo coordinates.
(new ymaps.Hint(myMap, {
  projection: ymaps.projection.wgs84Mercator,
  layout: ymaps.templateLayoutFactory.createClass('{{ content }}')
})).open(myMap.getCenter(), 'Hello');
```

#### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

#### Events

Name	Description
<a href="#">close</a>	Closing the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">open</a>	Opening the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .

#### Methods

Name	Returns	Description
<a href="#">close([force])</a>	<a href="#">vow.Promise</a>	Closes the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">getData()</a>		Returns info object data. Inherited from <a href="#">IPopup</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object to return the overlay. Inherited from <a href="#">IPopup</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>	Returns the overlay, if one exists. Inherited from <a href="#">IPopup</a> .

Name	Returns	Description
<a href="#">getPosition()</a>		Returns the coordinates of the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">isOpen()</a>	Boolean	Returns the info object state: open/closed. Inherited from <a href="#">IPopup</a> .
<a href="#">open([position[, data]])</a>	<a href="#">vow.Promise</a>	Opens the info object at the specified position. If the info object is already open, it moves it to the specified point. The format and content of the coordinates is determined by the <a href="#">IProjection</a> that is in the options. Inherited from <a href="#">IPopup</a> .
<a href="#">setData(data)</a>	<a href="#">vow.Promise</a>	Defines new data for the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">setPosition(position)</a>	<a href="#">vow.Promise</a>	Specifies a new position for the info object. Inherited from <a href="#">IPopup</a> .

## Hotspot

Extends [IHotspot](#).

Hotspot.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
Hotspot (shape[, zIndex])
```

Hotspot.

### Parameters:

Parameter	Default value	Description
<a href="#">shape</a> *	—	Type: <a href="#">IShape</a> The hotspot shape.
<a href="#">zIndex</a>	0	Type: Number zIndex of the hotspot.

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .

**Events**

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseleave</a>	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mousemove</a>	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseup</a>	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">multitouchend</a>	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface. Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getShape()</a>	<a href="#">IShape</a>	<p>Returns the hotspot shape.</p> <p>Inherited from <a href="#">IHotspot</a>.</p>
<a href="#">getZIndex()</a>	Number	<p>Returns <code>zIndex</code> of the hotspot.</p> <p>Inherited from <a href="#">IHotspot</a>.</p>
<a href="#">setShape(shape)</a>		<p>Sets the hotspot shape.</p> <p>Inherited from <a href="#">IHotspot</a>.</p>
<a href="#">setZIndex(zIndex)</a>		<p>Sets the z-index of the hotspot.</p> <p>Inherited from <a href="#">IHotspot</a>.</p>

## hotspot

### hotspot.layer

#### hotspot.layer.addon

#### hotspot.layer.addon.balloon

**Note:** The constructor of the `hotspot.layer.addon.balloon` class is hidden, as this class is not intended for autonomous initialization.

Static object.

#### [Methods](#)

#### Methods

Name	Returns	Description
<a href="#">get(layer)</a>	<a href="#">IPopupManager</a>	Returns manager of the balloon of a hotspot layer.

#### *Methods details*

#### get

```
{IPopupManager} get(layer)
```

**Returns** manager of the balloon of a hotspot layer.

#### Parameters:

Parameter	Default value	Description
<a href="#">layer</a> *	—	Type: <a href="#">hotspot.Layer</a> Hotspot layer.

\* Mandatory parameter/option.

#### Example:

```
ymaps.hotspot.layer.addon.balloon.get(layer)
```

#### hotspot.layer.addon.hint

**Note:** The constructor of the `hotspot.layer.addon.hint` class is hidden, as this class is not intended for autonomous initialization.

Static object.

#### [Methods](#)

#### Methods

Name	Returns	Description
<a href="#">get(layer)</a>	<a href="#">IPopupManager</a>	Returns manager of the hint on a hotspot layer.

*Methods details***get**

```
{IPopupManager} get(layer)
```

**Returns** manager of the hint on a hotspot layer.

**Parameters:**

Parameter	Default value	Description
<a href="#">layer</a> *	—	Type: <a href="#">hotspot.Layer</a> Hotspot layer.

\* Mandatory parameter/option.

**Example:**

```
ymaps.hotspot.layer.addon.hint.get(layer)
```

**hotspot.layer.Balloon**

Extends [IBalloonManager](#).

Manager of the balloon of a hotspot layer. Allows a geo object to manage the hotspot layer by opening it and hiding it. It uses the map balloon manager [map.Balloon](#). Hotspot layers contain an instance of this class available as `myHotspotLayer.balloon`. Don't create new instances of this class unless necessary.

**See** [Balloon hotspot.Layer.balloon](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
hotspot.layer.Balloon(hotspotLayer)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">hotspotLayer</a> *	—	Type: Object Hotspot layer.

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

## Events

Name	Description
<a href="#">autopanbegin</a>	<p>Start of automatic shifting of the map center initiated by the <code>autoPan</code> method. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>target</code> - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> <p>Inherited from <a href="#">IBalloonManager</a>.</p>
<a href="#">autopanend</a>	<p>End of automatic shifting of the map center initiated by the <code>autoPan</code> method. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>target</code> - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> <p>Inherited from <a href="#">IBalloonManager</a>.</p>
<a href="#">beforeuserclose</a>	<p>The event which precedes <a href="#">Balloon.event:userclose</a>. Allows you to cancel the user's action by calling the <code>preventDefault</code> method. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>target</code> - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> <p>Inherited from <a href="#">IBalloonManager</a>.</p>
<a href="#">close</a>	<p>Closing the info object. Names of fields available via <a href="#">Event.get</a>:</p> <ul style="list-style-type: none"> <li><code>target</code> - Reference to the object where the closing occurred.</li> </ul> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">open</a>	<p>Opening the info object. Names of fields available via <a href="#">Event.get</a>:</p> <ul style="list-style-type: none"> <li><code>target</code> - Reference to the object where the opening occurred.</li> </ul> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">userclose</a>	<p>Balloon closed by the user. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>target</code> - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> <p>Inherited from <a href="#">IBalloonManager</a>.</p>

## Methods

Name	Returns	Description
<a href="#">autoPan()</a>	<a href="#">vow.Promise</a>	<p>Moves the map so that the balloon is visible.</p> <p>Inherited from <a href="#">IBalloonManager</a>.</p>
<a href="#">close([force])</a>	<a href="#">vow.Promise</a>	<p>Closes the info object.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">destroy()</a>		<p>Disables the info object manager.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>

Name	Returns	Description
<a href="#">getData()</a>	Object null	Returns the data of the info object or 'null'. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOptions()</a>	<a href="#">IOptionManager</a>  null	Returns the options manager or 'null'. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object to return the overlay. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	Returns the overlay, if one exists. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getPosition()</a>	Number[] null	Returns the coordinates of the info object or 'null'. Inherited from <a href="#">IPopupManager</a> .
<a href="#">isOpen()</a>	Boolean	Returns the info object state: open/closed. Inherited from <a href="#">IPopupManager</a> .
<a href="#">open([position[, data[, options]])</a>	<a href="#">vow.Promise</a>	Opens the balloon at the specified position.
<a href="#">setData(data)</a>	<a href="#">vow.Promise</a>	Defines new data for the info object. Inherited from <a href="#">IPopupManager</a> .
<a href="#">setOptions(options)</a>	<a href="#">vow.Promise</a>	Defines new options for the info object. Inherited from <a href="#">IPopupManager</a> .
<a href="#">setPosition(position)</a>	<a href="#">vow.Promise</a>	Specifies a new position for the info object. Inherited from <a href="#">IPopupManager</a> .

## Methods details

### open

```
{vow.Promise} open([position[, data[, options]])
```

Opens the balloon at the specified position.

**Returns** Promise object.

**Parameters:**

Parameter	Default value	Description
<a href="#">position</a>	—	Type: Number[]  The coordinates of the balloon opening in the global pixel coordinates.
<a href="#">data</a>	—	Type: Object  Data.
<a href="#">options</a>	—	Type: Object  Options.

**hotspot.layer.Hint**

Extends [IHintManager](#).

Manager of the hint on a hotspot layer. Allows to manage the hint on a hotspot layer by opening it and hiding it. It uses the map hint manager [map.Hint](#) inside itself. Hotspot layers contain an instance of this class available as `myHotspotLayer.hint`. Don't create new instances of this class unless necessary.

See [Hint hotspot.Layer.hint](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
hotspot.layer.Hint(hotspotLayer)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">hotspotLayer</a> *	—	Type: Object  Hotspot layer.

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

**Events**

Name	Description
<a href="#">close</a>	Closing the info object. Names of fields available via <a href="#">Event.get</a> : <ul style="list-style-type: none"> <li>target - Reference to the object where the closing occurred.</li> </ul> Inherited from <a href="#">IPopupManager</a> .

Name	Description
<a href="#">open</a>	<p>Opening the info object. Names of fields available via <a href="#">Event.get</a>:</p> <ul style="list-style-type: none"> <li>target - Reference to the object where the opening occurred.</li> </ul> <p>Inherited from <a href="#">IPopupManager</a>.</p>

## Methods

Name	Returns	Description
<a href="#">close</a> ([force])	<a href="#">vow.Promise</a>	<p>Closes the info object.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">destroy</a> ()		<p>Disables the info object manager.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">getData</a> ()	Object null	<p>Returns the data of the info object or 'null'.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">getOptions</a> ()	<a href="#">IOptionManager</a>  null	<p>Returns the options manager or 'null'.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">getOverlay</a> ()	<a href="#">vow.Promise</a>	<p>Returns the promise object to return the overlay.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">getOverlaySync</a> ()	<a href="#">IOverlay</a>  null	<p>Returns the overlay, if one exists.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">getPosition</a> ()	Number[] null	<p>Returns the coordinates of the info object or 'null'.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">isOpen</a> ()	Boolean	<p>Returns the info object state: open/closed.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">open</a> ([position[, data[, options]]])	<a href="#">vow.Promise</a>	<p>Opens a hint in the specified position.</p>
<a href="#">setData</a> (data)	<a href="#">vow.Promise</a>	<p>Defines new data for the info object.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>

Name	Returns	Description
<a href="#">setOptions(options)</a>	<a href="#">vow.Promise</a>	Defines new options for the info object.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">setPosition(position)</a>	<a href="#">vow.Promise</a>	Specifies a new position for the info object.  Inherited from <a href="#">IPopupManager</a> .

## Methods details

### open

```
{vow.Promise} open([position[, data[, options]])
```

Opens a hint in the specified position.

**Returns** Promise object.

#### Parameters:

Parameter	Default value	Description
<a href="#">position</a>	—	Type: <a href="#">Number[]</a>  The coordinates of the balloon opening in the global pixel coordinates.
<a href="#">data</a>	—	Type: <a href="#">Object</a>  Data.
<a href="#">options</a>	—	Type: <a href="#">Object</a>  Options.

### hotspot.layer.Object

Extends [IHotspotLayerObject](#).

Object of the hotspot layer.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
hotspot.layer.Object(shape, feature, options)
```

Creates an object of the hotspot layer.

#### Parameters:

Parameter	Default value	Description
<a href="#">shape</a> *	—	Type: <a href="#">IShape</a>  The hotspot shape.

Parameter	Default value	Description
<code>feature</code> *	—	Type: Object  The description of the <a href="#">GeoObject</a> object.
<code>options</code> *	—	Type: Object  Object options.

\* Mandatory parameter/option.

### Fields

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<code>options</code>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

### Events

Name	Description
<code>click</code>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<code>contextmenu</code>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<code>dblclick</code>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<code>mousedown</code>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<code>mouseenter</code>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<code>mouseleave</code>	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape that defines a hotspot. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IHotspotLayerObject</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">getGeometry()</a>	Object	Returns the actual geometry of an object.  Inherited from <a href="#">IHotspotLayerObject</a> .
<a href="#">getHotspot()</a>	<a href="#">IHotspot</a>	Returns object describing a hotspot.  Inherited from <a href="#">IHotspotLayerObject</a> .
<a href="#">getId()</a>	Number	Returns object ID.  Inherited from <a href="#">IHotspotLayerObject</a> .
<a href="#">getProperties()</a>	Object	Returns object data.  Inherited from <a href="#">IHotspotLayerObject</a> .
<a href="#">setGeometry(geometry)</a>		Defines the actual geometry of the object.  Inherited from <a href="#">IHotspotLayerObject</a> .
<a href="#">setId(id)</a>		Sets the object ID.  Inherited from <a href="#">IHotspotLayerObject</a> .
<a href="#">setProperties(properties)</a>		Sets the object's data.  Inherited from <a href="#">IHotspotLayerObject</a> .

**hotspot.Layer**

Extends [IChildOnMap](#), [ICustomizable](#).

Hotspot layer.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
hotspot.Layer(objectSource[, options])
```

Creates the hotspot layer. Each individual area on the layer consists of an object - [hotspot.layer.Object](#).

**Parameters:**

Parameter	Default value	Description
<a href="#">objectSource</a> *	—	Type: <a href="#">IHotspotObjectSource</a>  Source of objects for the layer.

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object  Layer options. Options for balloons <a href="#">Balloon</a> and hints <a href="#">Hint</a> on the hotspot layer must be indicated with the "balloon" and "hint" prefixes.
<a href="#">options.cursor</a>	'pointer'	Type: String  Type of cursor.
<a href="#">options.dontChangeCursor</a>	false	Type: Boolean  Option that prevents changing the cursor when pointing at an object on the layer. By default, cursors change.
<a href="#">options.hasBalloon</a>	true	Type: Boolean  Flag showing whether the layer has balloons. If the flag value is "false", the ".balloon" field will not be created for the layer.
<a href="#">options.hasHint</a>	true	Type: Boolean  Flag showing whether the layer has hints. If the flag value is "false", the ".hint" field will not be created for the layer.
<a href="#">options.interactivityModel</a>	'default#layer'	Type: String  Interactivity model for the layer. Available keys and their values are listed in the description of <a href="#">interactivityModel.storage</a> .
<a href="#">options.openBalloonOnClick</a>	true	Type: Boolean  Option that prevents opening the balloon when clicking on a hotspot object. By default, opening the balloon is allowed.
<a href="#">options.openEmptyBalloon</a>	false	Type: String  Open the balloon with empty contents.
<a href="#">options.openEmptyHint</a>	false	Type: String  Show a popup hint with empty contents.
<a href="#">options.openHintOnHover</a>	true	Type: Boolean  Option that prevents showing the hint when pointing at a hotspot object. By default, showing hints is allowed.

Parameter	Default value	Description
<a href="#">options.pane</a>	'events'	Type: <a href="#">IEventPane</a>  Container with map elements that events are listened on.
<a href="#">options.zIndex</a>	—	Type: Number  The zIndex value of the layer. When searching for the active object on a single layer and the cursor falls on two objects simultaneously, the object with the larger zIndex value is selected. From that point on - for example, when determining which layer's shape takes priority - the zIndex value of the layer is used. Please note that the zIndex of the layer will determine the relative position of the objects on the layer, as well as single objects added to particular map panes. For example, if you want to place hotspot objects over the area objects, you should set zIndex=201 for the hotspot layer. See <a href="#">map.pane.Manager</a> .

\* Mandatory parameter/option.

#### Example:

```
// Creating the source for hotspot data. We aren't setting the key value,
// so the name of the handler function (padding jsonp in the request) is generated automatically.
var objectSource = new ymaps.hotspot.ObjectSource('tiles/?%c'),
    hotspotLayer = new ymaps.hotspot.Layer(objectSource, {
      zIndex: 100,
      // Allowing the balloon to be opened without content.
      showEmptyBalloon: true
    });
```

#### Fields

Name	Type	Description
<a href="#">balloon</a>	<a href="#">hotspot.layer.Balloon</a>	Hotspot layer balloon.
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">hint</a>	<a href="#">hotspot.layer.Hint</a>	Hotspot layer hint.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

#### Events

Name	Description
<a href="#">addtomap</a>	Layer added to the map.
<a href="#">balloonclose</a>	Closing the balloon. Instance of the <a href="#">Event</a> class.
<a href="#">balloonopen</a>	Opening the balloon on the hotspot layer. Instance of the <a href="#">Event</a> class.

Name	Description
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>activeObject</code> (<a href="#">hotspot.layer.Object</a>) - The object of the layer where the event occurred.</li><li>• <code>coords</code> - Geographical coordinates of the point at which the event occurred.</li><li>• <code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li><li>• <code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li><li>• <code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>activeObject</code> (<a href="#">hotspot.layer.Object</a>) - The object of the layer where the event occurred.</li><li>• <code>coords</code> - Geographical coordinates of the point at which the event occurred.</li><li>• <code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li><li>• <code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li><li>• <code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>activeObject</code> (<a href="#">hotspot.layer.Object</a>) - The object of the layer where the event occurred.</li><li>• <code>coords</code> - Geographical coordinates of the point at which the event occurred.</li><li>• <code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li><li>• <code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li><li>• <code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>
<a href="#">hintclose</a>	<p>Closing the hint. Instance of the <a href="#">Event</a> class.</p>
<a href="#">hintopen</a>	<p>Opening the hint on the hotspot layer. Instance of the <a href="#">Event</a> class.</p>

Name	Description
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>activeObject</code> (<a href="#">hotspot.layer.Object</a>) - The object of the layer where the event occurred.</li><li>• <code>coords</code> - Geographical coordinates of the point at which the event occurred.</li><li>• <code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li><li>• <code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li><li>• <code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>activeObject</code> (<a href="#">hotspot.layer.Object</a>) - The object of the layer where the event occurred.</li><li>• <code>coords</code> - Geographical coordinates of the point at which the event occurred.</li><li>• <code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li><li>• <code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li><li>• <code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>activeObject</code> (<a href="#">hotspot.layer.Object</a>) - The object of the layer where the event occurred.</li><li>• <code>coords</code> - Geographical coordinates of the point at which the event occurred.</li><li>• <code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li><li>• <code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li><li>• <code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>activeObject</code> (<a href="#">hotspot.layer.Object</a>) - The object of the layer where the event occurred.</li><li>• <code>coords</code> - Geographical coordinates of the point at which the event occurred.</li><li>• <code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li><li>• <code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li><li>• <code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>

Name	Description
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>activeObject</code> (<a href="#">hotspot.layer.Object</a>) - The object of the layer where the event occurred.</li><li>• <code>coords</code> - Geographical coordinates of the point at which the event occurred.</li><li>• <code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li><li>• <code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li><li>• <code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Names of fields that are available via the <code>IMultiTouchEvent#get</code> method:</p> <ul style="list-style-type: none"><li>• <code>activeObject</code> (<a href="#">hotspot.layer.Object</a>) - The object of the layer where the event occurred.</li><li>• <code>coords</code> - Geographical coordinates of the point at which the event occurred.</li><li>• <code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li><li>• <code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li><li>• <code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Names of fields that are available via the <code>IMultiTouchEvent#get</code> method:</p> <ul style="list-style-type: none"><li>• <code>activeObject</code> (<a href="#">hotspot.layer.Object</a>) - The object of the layer where the event occurred.</li><li>• <code>coords</code> - Geographical coordinates of the point at which the event occurred.</li><li>• <code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li><li>• <code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li><li>• <code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>

Name	Description
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Names of fields that are available via the <code>IMultiTouchEvent.get</code> method:</p> <ul style="list-style-type: none"> <li><code>activeObject</code> (<a href="#">hotspot.layer.Object</a>) - The object of the layer where the event occurred.</li> <li><code>coords</code> - Geographical coordinates of the point at which the event occurred.</li> <li><code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li> <li><code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li> <li><code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li> <li><code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul>
<a href="#">optionschange</a>	<p>Change to the object options. Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed. Data fields:</p> <ul style="list-style-type: none"> <li><code>oldParent</code> - Old parent.</li> <li><code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">removefrommap</a>	<p>Layer removed from the map.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>activeObject</code> (<a href="#">hotspot.layer.Object</a>) - The object of the layer where the event occurred.</li> <li><code>coords</code> - Geographical coordinates of the point at which the event occurred.</li> <li><code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li> <li><code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li> <li><code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li> <li><code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul>

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map.
<a href="#">getObjectInPosition(coords)</a>	<a href="#">vow.Promise</a>	Allows to get the layer object in the specified geo point at the current zoom level. The method call automatically downloads the tile which covers the point, if it is not yet downloaded. After that, the downloaded data will be used for searching.

Name	Returns	Description
<a href="#">getObjectsInPosition(coords)</a>	<a href="#">vow.Promise</a>	Allows to get the layer object in the specified geo point at the current zoom level. The method call automatically downloads the tile which covers the point, if it is not yet downloaded. After that, the downloaded data will be used for searching.
<a href="#">getObjectSource()</a>	<a href="#">IHotspotObjectSource</a>	Returns source of objects for the hotspot layer.
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IChildOnMap</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .
<a href="#">update()</a>		Updates the hotspot layer. After this command has been executed, previously loaded objects are removed from the container and new data is requested.

## Fields details

### balloon

```
{hotspot.layer.Balloon} balloon
```

Hotspot layer balloon.

### hint

```
{hotspot.layer.Hint} hint
```

Hotspot layer hint.

## Events details

### addtomap

Layer added to the map.

### balloonclose

Closing the balloon. Instance of the [Event](#) class.

### balloonopen

Opening the balloon on the hotspot layer. Instance of the [Event](#) class.

### click

Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#). Instance of the Event class. Names of fields that are available via the [Event.get](#) method:

- `activeObject` ([hotspot.layer.Object](#)) - The object of the layer where the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

### contextmenu

Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#). Instance of the Event class. Names of fields that are available via the [Event.get](#) method:

- `activeObject` ([hotspot.layer.Object](#)) - The object of the layer where the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

### dblclick

Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#). Instance of the Event class. Names of fields that are available via the [Event.get](#) method:

- `activeObject` ([hotspot.layer.Object](#)) - The object of the layer where the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

### hintclose

Closing the hint. Instance of the [Event](#) class.

### hintopen

Opening the hint on the hotspot layer. Instance of the [Event](#) class.

### mousedown

Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#). Instance of the Event class. Names of fields that are available via the [Event.get](#) method:

- `activeObject` ([hotspot.layer.Object](#)) - The object of the layer where the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

### mouseenter

Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#). Instance of the Event class. Names of fields that are available via the [Event.get](#) method:

- `activeObject` ([hotspot.layer.Object](#)) - The object of the layer where the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

### mouseleave

Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#). Instance of the Event class. Names of fields that are available via the [Event.get](#) method:

- `activeObject` ([hotspot.layer.Object](#)) - The object of the layer where the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

### mousemove

Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. Instance of the Event class. Names of fields that are available via the [Event.get](#) method:

- `activeObject` ([hotspot.layer.Object](#)) - The object of the layer where the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

### mouseup

Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#). Instance of the Event class. Names of fields that are available via the [Event.get](#) method:

- `activeObject` ([hotspot.layer.Object](#)) - The object of the layer where the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

### multitouchend

End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the `IMultiTouchEvent` interface with information about touches. Names of fields that are available via the `IMultiTouchEvent#get` method:

- `activeObject` ([hotspot.layer.Object](#)) - The object of the layer where the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.

- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

### **multitouchmove**

Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the `IMultiTouchEvent` interface with information about touches. Names of fields that are available via the `IMultiTouchEvent#get` method:

- `activeObject` ([hotspot.layer.Object](#)) - The object of the layer where the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

### **multitouchstart**

Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the `IMultiTouchEvent` interface with information about touches. Names of fields that are available via the `IMultiTouchEvent.get` method:

- `activeObject` ([hotspot.layer.Object](#)) - The object of the layer where the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

### **removefrommap**

Layer removed from the map.

### **wheel**

Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#). Instance of the `Event` class. Names of fields that are available via the `Event.get` method:

- `activeObject` ([hotspot.layer.Object](#)) - The object of the layer where the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

## **Methods details**

### **getMap**

```
{Map} getMap()
```

**Returns** reference to the map.

### **getObjectInPosition**

```
{vow.Promise} getObjectInPosition(coords)
```

Allows to get the layer object in the specified geo point at the current zoom level. The method call automatically downloads the tile which covers the point, if it is not yet downloaded. After that, the downloaded data will be used for searching.

**Returns** the Promise object that will be resolved by the layer object covering the point. If the point falls on several layer objects, the object with the maximum zIndex will be returned.

**Parameters:**

Parameter	Default value	Description
<code>coords *</code>	—	Type: Number[]  The geo coordinates of the point.

\* Mandatory parameter/option.

### getObjectsInPosition

```
{vow.Promise} getObjectsInPosition(coords)
```

Allows to get the layer object in the specified geo point at the current zoom level. The method call automatically downloads the tile which covers the point, if it is not yet downloaded. After that, the downloaded data will be used for searching.

**Returns** the Promise object that will be resolved by an array of layer objects covering the point.

**Parameters:**

Parameter	Default value	Description
<code>coords *</code>	—	Type: Number[]  The geo coordinates of the point.

\* Mandatory parameter/option.

### getObjectSource

```
{IHotspotObjectSource} getObjectSource()
```

**Returns** source of objects for the hotspot layer.

**Example:**

```
// Updating the data source for the hotspot layer.
hotspotLayer.getObjectSource().setTileUrlTemplate('newSource/??c');
hotspotLayer.update();
```

### update

```
{ } update()
```

Updates the hotspot layer. After this command has been executed, previously loaded objects are removed from the container and new data is requested.

## hotspot.ObjectSource

The standard implementation of the [IHotspotObjectSource](#) interface.

**Note:** Deprecated: This module is supported in version 2.1 for backward compatibility. Use the `hotspot.layer.ObjectSource` module.

See `hotspot.layer.ObjectSource`

[Constructor](#) | [Methods](#)

### Constructor

```
hotspot.ObjectSource()
```

### Methods

Name	Returns	Description
<a href="#">getKey(tileNumber, zoom)</a>	String	Returns the name of the callback function (padding) in the jsonp request if it is static, or null if a dynamic jsonp callback can be used. Templates support the same substitutions as in the template for the tile URL. All characters other than letters, numbers, and the "_" symbol will be replaced with "_".
<a href="#">getKeyTemplate()</a>	String	Returns template for the tile ID.
<a href="#">getTileUrl(tileNumber, zoom)</a>	String	Returns the URL of data for a specific tile.
<a href="#">getTileUrlTemplate()</a>	String	Returns template for the URL for tile data.
<a href="#">parseResponse(layer, res, callback, tileNumber, zoom)</a>		Parses the server response. Sends the callback an array of hotspot objects generated using the json description.
<a href="#">restrict(layer, tileNumber, zoom)</a>	Boolean	Method that is called before sending a request for tile data. If the method returns true, the request will not be sent to the server, and an empty array of objects will be returned as the response. The standard implementation of the method checks whether the "zoom" parameter is within the limits of [options.minZoom, options.maxZoom]. It also checks whether the center of the requested tile falls within the specified options.bounds. If options.bounds=null, this parameter is not checked.
<a href="#">setKeyTemplate(template)</a>		Sets a new tile ID template.
<a href="#">setTileUrlTemplate(template)</a>		Sets a new template for the tile data URL.

## Methods details

### getKey

```
{String} getKey(tileNumber, zoom)
```

Returns the name of the callback function (padding) in the jsonp request if it is static, or null if a dynamic jsonp callback can be used. Templates support the same substitutions as in the template for the tile URL. All characters other than letters, numbers, and the "\_" symbol will be replaced with "\_".

**Returns** tile ID. Used when creating padding in the jsonp request for data.

#### Parameters:

Parameter	Default value	Description
<code>tileNumber *</code>	—	Type: Number[] Tile number (tile coordinates).
<code>zoom *</code>	—	Type: Integer Zoom level.

\* Mandatory parameter/option.

### getKeyTemplate

```
{String} getKeyTemplate()
```

**Returns** template for the tile ID.

### getTileUrl

```
{String} getTileUrl(tileNumber, zoom)
```

**Returns** the URL of data for a specific tile.

#### Parameters:

Parameter	Default value	Description
<code>tileNumber *</code>	—	Type: Number[] Tile number (tile coordinates).
<code>zoom *</code>	—	Type: Integer Zoom level.

\* Mandatory parameter/option.

#### Example:

```
var hotspotObjectSource = new ymaps.hotspot.ObjectSource('dataSource/%c');
hotspotObjectSource.getTileUrl = function (tileNumber, zoom) {
  if (zoom > 10) {
    // For large scales, use the provided data path.
    return ymaps.hotspot.ObjectSource.prototype.call(this, tileNumber, zoom);
  } else {
    // For small scales, use a different path.
    return 'otherSource/getHotspots.xml?z=' + zoom + '&codeph&codeph&x=' + tileNumber[0] +
      '&codeph&codeph&y=' + tileNumber[1];
  }
};
```

**getTileUrlTemplate**

```
{String} getTileUrlTemplate()
```

**Returns** template for the URL for tile data.

**parseResponse**

```
{ } parseResponse(layer, res, callback, tileNumber, zoom)
```

Parses the server response. Sends the callback an array of hotspot objects generated using the json description.

**Parameters:**

Parameter	Default value	Description
<code>layer *</code>	—	Type: <code>hotspot.Layer</code> The layer that the objects belong to.
<code>res *</code>	—	Type: Object Server response.
<code>callback *</code>	—	Type: Function Handler function.
<code>tileNumber *</code>	—	Type: Number[] Number of the tile that the response is for.
<code>zoom *</code>	—	Type: Number The zoom level that the response is for; array of objects.

\* Mandatory parameter/option.

**restrict**

```
{Boolean} restrict(layer, tileNumber, zoom)
```

Method that is called before sending a request for tile data. If the method returns true, the request will not be sent to the server, and an empty array of objects will be returned as the response. The standard implementation of the method checks whether the "zoom" parameter is within the limits of [options.minZoom, options.maxZoom]. It also checks whether the center of the requested tile falls within the specified options.bounds. If options.bounds=null, this parameter is not checked.

**Returns** true if the tile extends beyond the boundaries of the data area (there is no data for this tile), or false if it does not (there is data).

**Parameters:**

Parameter	Default value	Description
<code>layer *</code>	—	Type: <code>hotspot.Layer</code> Hotspot layer.

Parameter	Default value	Description
<code>tileNumber</code> *	—	Type: Number[] Tile number.
<code>zoom</code> *	—	Type: Integer Zoom level.

\* Mandatory parameter/option.

#### Example:

```
// Example of redefining the "restrict" method
// Let's assume there is only data for Murmansk and Novosibirsk.
var myMap = new ymaps.Map('map', {center: [32.5, 68.9] , zoom: 9});
var geoBounds = [
  [[31.729958, 69.369182], [34.203324, 68.666473]], // Murmansk
  [[82.179084, 55.341085], [83.725642, 54.670738]] // Novosibirsk
];
var projection = myMap.options.get('projection');
var myHotspotSource = new ymaps.hotspot.ObjectSource('http://www.myDomain.ru/tiles/?%c', '%c');

myHotspotSource.restrict = function(layer, tileNumber, zoom) {
  // Calculating the pixel boundaries of the cities for this zoom level.
  var boundsFromPoints = ymaps.util.bounds.fromPoints;
  var toGlobalPixels = projection.toGlobalPixels;
  var pixelBounds = [
    boundsFromPoints(
      toGlobalPixels(geoBounds[0][0]), zoom),
      toGlobalPixels(geoBounds[0][1]), zoom)
    ),
    boundsFromPoints(
      toGlobalPixels(geoBounds[1][0]), zoom),
      toGlobalPixels(geoBounds[1][1]), zoom)
    )
  ];

  // Calculating the pixel boundaries of the tile
  var leftTop = [tileNumber[0] * 256, tileNumber[1] * 256];
  var tileBounds = [leftTop, [leftTop[0] + 256, leftTop[1] + 256]];
  var intersects = ymaps.util.bounds.intersects;
  // If the tile's pixel boundaries intersect with the pixel boundaries of the specified areas,
  // we have to send a request for data
  if (intersects(pixelBounds[0], tileBounds) || (intersects(pixelBounds[1], tileBounds)) {
    return false;
  }

  // For all the other tiles, this source doesn't have any data.
  return true;
}
```

#### setKeyTemplate

```
{ } setKeyTemplate(template)
```

Sets a new tile ID template.

#### Parameters:

Parameter	Default value	Description
<code>template</code> *	—	Type: String Template for the ID.

\* Mandatory parameter/option.

#### setTileUrlTemplate

```
{ } setTileUrlTemplate(template)
```

Sets a new template for the tile data URL.

**Parameters:**

Parameter	Default value	Description
<a href="#">template</a> *	—	Type: String URL template

\* Mandatory parameter/option.

## interactivityModel

### interactivityModel.storage

Static object.

Instance of [util.Storage](#)

Storage for the interactivity model. Interactivity models allow objects to handle DOM events in different ways. Available interactivity model keys:

- 'default#opaque' - The object generates all DOM events and does not throw them to the map. Map behaviors will not work when hovering over or clicking on objects with this interactivity model.
- 'default#geoObject' - The object generates all DOM events. The events "wheel", "mousedown", "dblclick", "contextmenu", "multitouchstart", "multitouchmove" and "multitouchend" are thrown to the map. If the "scrollZoom", "dblClickZoom" or "magnifier" behaviors are enabled on the map, they will work via objects with this interactivity model, in contrast to the objects with the "default#opaque" model.
- 'default#layer' - The object generates all DOM events. The events "wheel", "mousedown", "contextment", "multitouchstart", "multitouchmove", and "multitouchend" are thrown to the map. If the "scrollZoom", "drag", or "magnifier" behaviors are enabled on the map, they will work via objects with this interactivity model.
- 'default#transparent' - The object generates all DOM events, then throws them to the map.
- 'default#silent' - The object does not generate all DOM events but throws them to the map.

#### Methods

#### Methods

Name	Returns	Description
<a href="#">add(key, object)</a>	<a href="#">util.Storage</a>	Adds an object to storage.
<a href="#">get(key)</a>	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
<a href="#">remove(key)</a>	<a href="#">util.Storage</a>	Deletes the "key: value" pair from storage.

## Interfaces

### IBalloon

Extends [IPopup](#).

Interface for a balloon.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

IBalloon()

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

**Events**

Name	Description
<a href="#">close</a>	Closing the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">open</a>	Opening the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .

**Methods**

Name	Returns	Description
<a href="#">autoPan()</a>	<a href="#">vow.Promise</a>	Moves the map so that the balloon is visible.
<a href="#">close([force])</a>	<a href="#">vow.Promise</a>	Closes the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">getData()</a>		Returns info object data. Inherited from <a href="#">IPopup</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object to return the overlay. Inherited from <a href="#">IPopup</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>	Returns the overlay, if one exists. Inherited from <a href="#">IPopup</a> .
<a href="#">getPosition()</a>		Returns the coordinates of the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">isOpen()</a>	Boolean	Returns the info object state: open/closed. Inherited from <a href="#">IPopup</a> .

Name	Returns	Description
<code>open([position[, data]])</code>	<code>vow.Promise</code>	Opens the info object at the specified position. If the info object is already open, it moves it to the specified point. The format and content of the coordinates is determined by the <a href="#">IProjection</a> that is in the options.  Inherited from <a href="#">IPopup</a> .
<code>setData(data)</code>	<code>vow.Promise</code>	Defines new data for the info object.  Inherited from <a href="#">IPopup</a> .
<code>setPosition(position)</code>	<code>vow.Promise</code>	Specifies a new position for the info object.  Inherited from <a href="#">IPopup</a> .

### Methods details

#### autoPan

```
{vow.Promise} autoPan()
```

Moves the map so that the balloon is visible.

**Returns** Promise object.

## IBalloonLayout

Extends [ILayout](#).

Interface for the balloon layout.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IBalloonLayout()
```

### Fields

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IDomEventEmitter</a> .

### Events

Name	Description
<code>click</code>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">emptinesschange</a>	<p>Change to the empty layout indicator. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">parentelementchange</a>	<p>Change to the parent element. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the layout. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">userclose</a>	<p>The Close button is selected. The user closes the balloon. Implements the <a href="#">IEvent</a> interface.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		<p>Destructor. Called when activity with the layout is finished.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">getData()</a>	Object	<p>Returns layout data object.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">getParentElement()</a>	HTMLElement	<p>Returns parent HTML element.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	<p>Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element.</p> <p>Inherited from <a href="#">ILayout</a>.</p>

Name	Returns	Description
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.  Inherited from <a href="#">ILayout</a> .
<a href="#">setData(data)</a>		Sets layout data.  Inherited from <a href="#">ILayout</a> .
<a href="#">setParentElement(parent)</a>		Adds the layout to the DOM tree.  Inherited from <a href="#">ILayout</a> .

### Events details

#### userclose

The Close button is selected. The user closes the balloon. Implements the [IEvent](#) interface.

## IBalloonManager

Extends [IPopupManager](#).

Interface for the balloon manager.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IBalloonManager()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

### Events

Name	Description
<a href="#">autopanbegin</a>	Start of automatic shifting of the map center initiated by the autoPan method. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>target - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul>
<a href="#">autopanend</a>	End of automatic shifting of the map center initiated by the autoPan method. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>target - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul>

Name	Description
<a href="#">beforeuserclose</a>	The event which precedes <a href="#">Balloon.event:userclose</a> . Allows you to cancel the user's action by calling the <code>preventDefault</code> method. Instance of the <code>Event</code> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>target</code> - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul>
<a href="#">close</a>	Closing the info object. Names of fields available via <a href="#">Event.get</a> : <ul style="list-style-type: none"> <li><code>target</code> - Reference to the object where the closing occurred.</li> </ul> Inherited from <a href="#">IPopupManager</a> .
<a href="#">open</a>	Opening the info object. Names of fields available via <a href="#">Event.get</a> : <ul style="list-style-type: none"> <li><code>target</code> - Reference to the object where the opening occurred.</li> </ul> Inherited from <a href="#">IPopupManager</a> .
<a href="#">userclose</a>	Balloon closed by the user. Instance of the <code>Event</code> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>target</code> - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul>

## Methods

Name	Returns	Description
<a href="#">autoPan()</a>	<a href="#">vow.Promise</a>	Moves the map so that the balloon is visible.
<a href="#">close([force])</a>	<a href="#">vow.Promise</a>	Closes the info object. Inherited from <a href="#">IPopupManager</a> .
<a href="#">destroy()</a>		Disables the info object manager. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getData()</a>	<code>Object null</code>	Returns the data of the info object or 'null'. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOptions()</a>	<a href="#">IOptionManager null</a>	Returns the options manager or 'null'. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object to return the overlay. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay null</a>	Returns the overlay, if one exists. Inherited from <a href="#">IPopupManager</a> .

Name	Returns	Description
<a href="#">getPosition()</a>	Number[] null	Returns the coordinates of the info object or 'null'.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">isOpen()</a>	Boolean	Returns the info object state: open/closed.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">open([position[, data[, options]])</a>	<a href="#">vow.Promise</a>	Opens the info object at the specified position.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">setData(data)</a>	<a href="#">vow.Promise</a>	Defines new data for the info object.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">setOptions(options)</a>	<a href="#">vow.Promise</a>	Defines new options for the info object.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">setPosition(position)</a>	<a href="#">vow.Promise</a>	Specifies a new position for the info object.  Inherited from <a href="#">IPopupManager</a> .

## Events details

### autopanbegin

Start of automatic shifting of the map center initiated by the autoPan method. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- target - Reference to the [IBalloonOwner](#) object.

### autopanend

End of automatic shifting of the map center initiated by the autoPan method. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- target - Reference to the [IBalloonOwner](#) object.

### beforeuserclose

The event which precedes [Balloon.event:userclose](#). Allows you to cancel the user's action by calling the preventDefault method. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- target - Reference to the [IBalloonOwner](#) object.

### userclose

Balloon closed by the user. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- target - Reference to the [IBalloonOwner](#) object.

## Methods details

### autoPan

```
{vow.Promise} autoPan()
```

Moves the map so that the balloon is visible.

**Returns** Promise object.

## IBalloonOwner

An object with a balloon, which can be accessed through the "balloon" property.

[Constructor](#) | [Fields](#) | [Events](#)

### Constructor

```
IBalloonOwner()
```

### Fields

Name	Type	Description
<a href="#">balloon</a>	<a href="#">IBalloonManager</a>	Balloon for an object.

### Events

Name	Description
<a href="#">balloonclose</a>	Closing the balloon.
<a href="#">balloonopen</a>	Opening the balloon.

### Fields details

#### balloon

```
{IBalloonManager} balloon
```

Balloon for an object.

### Events details

#### balloonclose

Closing the balloon.

#### balloonopen

Opening the balloon.

## IBaseCircleGeometry

Extends [IBaseGeometry](#), [ICircleGeometryAccess](#).

Interface for the "Circle" geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
IBaseCircleGeometry()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

**Events**

Name	Description
<a href="#">change</a>	Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>oldCoordinates</code> - Old coordinates of the center.</li> <li><code>newCoordinates</code> - New coordinates of the center.</li> <li><code>oldRadius</code> - Old radius.</li> <li><code>newRadius</code> - New radius.</li> </ul> Inherited from <a href="#">ICircleGeometryAccess</a> .

**Methods**

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the circle.  Inherited from <a href="#">ICircleGeometryAccess</a> .
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	Switches the object to "frozen" mode.  Inherited from <a href="#">IFreezable</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for a point on the circle closest to the anchorPosition.  Inherited from <a href="#">ICircleGeometryAccess</a> .

Name	Returns	Description
<a href="#">getCoordinates()</a>	Number[][] null	Returns coordinates of the center of the circle.  Inherited from <a href="#">ICircleGeometryAccess</a> .
<a href="#">getRadius()</a>	Number	Returns radius of the circle.  Inherited from <a href="#">ICircleGeometryAccess</a> .
<a href="#">getType()</a>	String	Returns the "Circle" string.
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false.  Inherited from <a href="#">IFreezable</a> .
<a href="#">setCoordinates(coordinates)</a>	<a href="#">ICircleGeometryAccess</a>	Sets the coordinates of the center of the circle.  Inherited from <a href="#">ICircleGeometryAccess</a> .
<a href="#">setRadius(radius)</a>	<a href="#">ICircleGeometryAccess</a>	Sets the radius of the circle.  Inherited from <a href="#">ICircleGeometryAccess</a> .
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode.  Inherited from <a href="#">IFreezable</a> .

## Methods details

### getType

```
{String} getType()
```

Returns the "Circle" string.

## IBaseGeometry

Extends [IEventEmitter](#).

Interface of a basic geometry.

[Constructor](#) | [Fields](#) | [Methods](#)

### Constructor

```
IBaseGeometry()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

**Methods**

Name	Returns	Description
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.
<a href="#">getType()</a>	String	Returns ID of the geometry type.

**Methods details****getBounds**

```
{Number[][]|null} getBounds()
```

**Returns** coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.

**getType**

```
{String} getType()
```

**Returns** ID of the geometry type.

**IBaseLinearRingGeometry**

Extends [IBaseGeometry](#), [ILinearRingGeometryAccess](#).

Interface for the "Closed contour" geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
IBaseLinearRingGeometry()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

**Events**

Name	Description
<a href="#">change</a>	<p>Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>oldCoordinates - Old coordinates</li> <li>newCoordinates - New coordinates.</li> <li>oldFillRule - Old fill rule.</li> <li>newFillRule - New fill rule.</li> </ul> <p>Inherited from <a href="#">ILinearRingGeometryAccess</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	<p>Checks whether the passed point is located inside the contour.</p> <p>Inherited from <a href="#">ILinearRingGeometryAccess</a>.</p>
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	<p>Switches the object to "frozen" mode.</p> <p>Inherited from <a href="#">IFreezable</a>.</p>
<a href="#">get(index)</a>	Number[]	<p>Returns coordinates of the point with the specified index.</p> <p>Inherited from <a href="#">ILinearRingGeometryAccess</a>.</p>
<a href="#">getBounds()</a>	Number[][] null	<p>Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.</p> <p>Inherited from <a href="#">IBaseGeometry</a>.</p>
<a href="#">getChildGeometry(index)</a>	<a href="#">IPointGeometryAccess</a>	<p>Creates and returns an <a href="#">IPointGeometryAccess</a> object for the specified contour on the polyline.</p> <p>Inherited from <a href="#">ILinearRingGeometryAccess</a>.</p>
<a href="#">getClosest(anchorPosition)</a>	Object	<p>Searches for a point on the contour closest to the anchorPosition.</p> <p>Inherited from <a href="#">ILinearRingGeometryAccess</a>.</p>

Name	Returns	Description
<a href="#">getCoordinates()</a>	Number[][]	Returns an array of geometry coordinates. Inherited from <a href="#">ILinearRingGeometryAccess</a> .
<a href="#">getFillRule()</a>	String	Returns ID of the fill rule. Inherited from <a href="#">ILinearRingGeometryAccess</a> .
<a href="#">getLength()</a>	Integer	Returns the number of points in the geometry. Inherited from <a href="#">ILinearRingGeometryAccess</a> .
<a href="#">getType()</a>	String	Returns the "LinearRing" string.
<a href="#">insert(index, coordinates)</a>	<a href="#">ILinearRingGeometryAccess</a>	Adds a new point with the specified index. Inherited from <a href="#">ILinearRingGeometryAccess</a> .
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from <a href="#">IFreezable</a> .
<a href="#">remove(index)</a>	Number[]	Removes the point with the specified index. Inherited from <a href="#">ILinearRingGeometryAccess</a> .
<a href="#">set(index, coordinates)</a>	<a href="#">ILinearRingGeometryAccess</a>	Sets coordinates of the point with the specified index. Inherited from <a href="#">ILinearRingGeometryAccess</a> .
<a href="#">setCoordinates(coordinates)</a>	<a href="#">ILinearRingGeometryAccess</a>	Sets an array of geometry coordinates. Inherited from <a href="#">ILinearRingGeometryAccess</a> .
<a href="#">setFillRule(fillRule)</a>	<a href="#">ILinearRingGeometryAccess</a>	Sets the contour fill rule. Inherited from <a href="#">ILinearRingGeometryAccess</a> .
<a href="#">splice(index, number)</a>	Number[][]	Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter. Inherited from <a href="#">ILinearRingGeometryAccess</a> .

Name	Returns	Description
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode. Inherited from <a href="#">IFreezable</a> .

### Methods details

#### getType

```
{String} getType()
```

Returns the "LinearRing" string.

## IBaseLineStringGeometry

Extends [IBaseGeometry](#), [ILineStringGeometryAccess](#).

Interface for the "Polyline" geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IBaseLineStringGeometry()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

### Events

Name	Description
<a href="#">change</a>	Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>oldCoordinates</code> - Old coordinates</li> <li><code>newCoordinates</code> - New coordinates.</li> </ul> Inherited from <a href="#">ILineStringGeometryAccess</a> .

### Methods

Name	Returns	Description
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	Switches the object to "frozen" mode. Inherited from <a href="#">IFreezable</a> .
<a href="#">get(index)</a>	Number[]	Returns coordinates of the point with the specified index. Inherited from <a href="#">ILineStringGeometryAccess</a> .

Name	Returns	Description
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getChildGeometry(index)</a>	<a href="#">IPointGeometryAccess</a>	Creates and returns an <a href="#">IPointGeometryAccess</a> object for the specified vertex on the polyline.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for a point on the polyline closest to the anchorPosition.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">getCoordinates()</a>	Number[][]	Returns an array of geometry coordinates.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">getLength()</a>	Integer	Returns the number of points in the geometry.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">getType()</a>	String	Returns the "LineString" string.
<a href="#">insert(index, coordinates)</a>	<a href="#">ILineStringGeometryAccess</a>	Adds a new point with the specified index.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false.  Inherited from <a href="#">IFreezable</a> .
<a href="#">remove(index)</a>	Number[]	Removes the point with the specified index.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">set(index, coordinates)</a>	<a href="#">ILineStringGeometryAccess</a>	Sets coordinates of the point with the specified index.  Inherited from <a href="#">ILineStringGeometryAccess</a> .

Name	Returns	Description
<a href="#">setCoordinates(coordinates)</a>	<a href="#">ILineStringGeometryAccess</a>	Sets an array of geometry coordinates.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">splice(index, number)</a>	Number[][]	Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode.  Inherited from <a href="#">IFreezable</a> .

### Methods details

#### getType

```
{String} getType()
```

Returns the "LineString" string.

## IBasePointGeometry

Extends [IBaseGeometry](#), [IPointGeometryAccess](#).

Interface for the "Point" geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IBasePointGeometry()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

### Events

Name	Description
<a href="#">change</a>	Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>oldCoordinates</code> - Old coordinates</li> <li><code>newCoordinates</code> - New coordinates.</li> </ul> Inherited from <a href="#">IPointGeometryAccess</a> .

**Methods**

Name	Returns	Description
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getCoordinates()</a>	Number[][] null	Returns coordinates of a point.  Inherited from <a href="#">IPointGeometryAccess</a> .
<a href="#">getType()</a>	String	Returns the "Point" string.
<a href="#">setCoordinates(coordinates)</a>	<a href="#">IPointGeometryAccess</a>	Sets coordinates of a point.  Inherited from <a href="#">IPointGeometryAccess</a> .

**Methods details****getType**

```
{String} getType()
```

Returns the "Point" string.

**IBasePolygonGeometry**

Extends [IBaseGeometry](#), [IPolygonGeometryAccess](#).

Interface for the "Polygon" geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
IBasePolygonGeometry()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

**Events**

Name	Description
<a href="#">change</a>	<p>Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>oldCoordinates - Old coordinates</li> <li>newCoordinates - New coordinates.</li> <li>oldFillRule - Old fill rule.</li> <li>newFillRule - New fill rule.</li> </ul> <p>Inherited from <a href="#">IPolygonGeometryAccess</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	<p>Checks whether the passed point is located inside the polygon.</p> <p>Inherited from <a href="#">IPolygonGeometryAccess</a>.</p>
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	<p>Switches the object to "frozen" mode.</p> <p>Inherited from <a href="#">IFreezable</a>.</p>
<a href="#">get(index)</a>	Number[][]	<p>Returns coordinates of the contour with the specified index.</p> <p>Inherited from <a href="#">IPolygonGeometryAccess</a>.</p>
<a href="#">getBounds()</a>	Number[][] null	<p>Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.</p> <p>Inherited from <a href="#">IBaseGeometry</a>.</p>
<a href="#">getChildGeometry(index)</a>	<a href="#">ILinearRingGeometryAccess</a>	<p>Creates and returns an <a href="#">ILinearRingGeometryAccess</a> object for the specified contour.</p> <p>Inherited from <a href="#">IPolygonGeometryAccess</a>.</p>
<a href="#">getClosest(anchorPosition)</a>	Object	<p>Searches for the point nearest to "anchorPosition" on the polygon contour.</p> <p>Inherited from <a href="#">IPolygonGeometryAccess</a>.</p>

Name	Returns	Description
<a href="#">getCoordinates()</a>	Number[][]	Returns an array of geometry coordinates. Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">getFillRule()</a>	String	Returns ID of the fill rule. Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">getLength()</a>	Integer	Returns the number of contours in the geometry. Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">getType()</a>	String	Returns the "Polygon" string.
<a href="#">insert(index, path)</a>	<a href="#">IPolygonGeometryAccess</a>	Adds a new contour with the specified index. Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from <a href="#">IFreezable</a> .
<a href="#">remove(index)</a>	<a href="#">ILinearRingGeometryAccess</a>	Removes the contour with the specified index. Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">set(index, path)</a>	<a href="#">IPolygonGeometryAccess</a>	Sets coordinates of the contour with the specified index. Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">setCoordinates(coordinates)</a>	<a href="#">IPolygonGeometryAccess</a>	Sets an array of geometry coordinates. Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">setFillRule(fillRule)</a>	<a href="#">IPolygonGeometryAccess</a>	Sets the polygon's fill rule. Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">splice(index, number)</a>	<a href="#">ILinearRingGeometryAccess[]</a>	Deletes a defined number of contours, starting from the specified index. New contours can be added in place of the deleted ones. Coordinates of the new contours can be passed as additional arguments after the "number" parameter. Inherited from <a href="#">IPolygonGeometryAccess</a> .

Name	Returns	Description
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode.  Inherited from <a href="#">IFreezable</a> .

### Methods details

#### getType

```
{String} getType()
```

Returns the "Polygon" string.

## IBaseRectangleGeometry

Extends [IBaseGeometry](#), [IRectangleGeometryAccess](#).

Interface for the "Rectangle" geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IBaseRectangleGeometry()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

### Events

Name	Description
<a href="#">change</a>	Change to corner coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>oldCoordinates</code> - Old coordinates of the corners.</li> <li><code>newCoordinates</code> - New coordinates of the corners.</li> </ul> Inherited from <a href="#">IRectangleGeometryAccess</a> .

### Methods

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the rectangle.  Inherited from <a href="#">IRectangleGeometryAccess</a> .
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	Switches the object to "frozen" mode.  Inherited from <a href="#">IFreezable</a> .

Name	Returns	Description
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for the point nearest to "anchorPosition" on the rectangle contour.  Inherited from <a href="#">IRectangleGeometryAccess</a> .
<a href="#">getCoordinates()</a>	Number[][]	Returns coordinates of two opposite corners of the rectangle.  Inherited from <a href="#">IRectangleGeometryAccess</a> .
<a href="#">getType()</a>	String	Returns the "Rectangle" string.
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false.  Inherited from <a href="#">IFreezable</a> .
<a href="#">setCoordinates(coordinates)</a>	<a href="#">IRectangleGeometryAccess</a>	Sets the coordinates of two opposite corners of the rectangle.  Inherited from <a href="#">IRectangleGeometryAccess</a> .
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode.  Inherited from <a href="#">IFreezable</a> .

## Methods details

### getType

```
{String} getType()
```

Returns the "Rectangle" string.

## IBehavior

Extends [IChildOnMap](#), [ICustomizable](#).

Map behavior. Adds a map reaction to certain user actions (such as dragging, right-click zooming, or multitouch). The default behavior is disabled and can be enabled using the "enable" method.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
IBehavior([options])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object Behavior options.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

**Events**

Name	Description
<a href="#">disable</a>	Disabling behaviors.
<a href="#">enable</a>	Enabling behaviors.
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li><code>oldParent</code> - Old parent.</li> <li><code>newParent</code> - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .

**Methods**

Name	Returns	Description
<a href="#">disable()</a>		Disables the behavior.
<a href="#">enable()</a>		Enables the behavior.
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">isEnabled()</a>	Boolean	Checks whether the behavior is enabled.

Name	Returns	Description
<code>setParent(parent)</code>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .

### Events details

#### disable

Disabling behaviors.

#### enable

Enabling behaviors.

### Methods details

#### disable

```
{ } disable()
```

Disables the behavior.

#### enable

```
{ } enable()
```

Enables the behavior.

#### isEnabled

```
{Boolean} isEnabled()
```

Checks whether the behavior is enabled.

**Returns** true if the behavior is enabled, otherwise false.

## ICanvasTile

Extends [ITile](#).

Interface for tiles that can be displayed in the canvas object's 2d context.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
ICanvasTile(url)
```

### Parameters:

Parameter	Default value	Description
<code>url</code> *	—	Type: String Tile URL.

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

**Events**

Name	Description
<a href="#">ready</a>	Tile ready event. Inherited from <a href="#">ITile</a> .

**Methods**

Name	Returns	Description
<a href="#">destroy()</a>		Destroys the tile. Inherited from <a href="#">ITile</a> .
<a href="#">isReady()</a>	Boolean	Checks tile readiness. Inherited from <a href="#">ITile</a> .
<a href="#">renderAt(context, canvasSize, bounds[, animate])</a>		Draws an image tile in the canvas object's 2d context.

**Methods details****renderAt**

```
{ } renderAt(context, canvasSize, bounds[, animate])
```

Draws an image tile in the canvas object's 2d context.

**Parameters:**

Parameter	Default value	Description
<a href="#">context</a> *	—	Type: Object  2D context of the canvas object.
<a href="#">canvasSize</a> *	—	Type: Number[]  Dimensions of the canvas HTML element.
<a href="#">bounds</a> *	—	Type: Number[][]  Area in client coordinates where the tile should be drawn.
<a href="#">animate</a>	false	Type: Boolean  If true, rendering is animated; if false, it is not.

\* Mandatory parameter/option.

## IChild

Extends [IEventEmitter](#).

Interface for an object that has a parent.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IChild()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

### Events

Name	Description
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"><li>• <code>oldParent</code> - Old parent.</li><li>• <code>newParent</code> - New parent.</li></ul>

### Methods

Name	Returns	Description
<a href="#">getParent()</a>	Object	Returns reference to the parent object.
<a href="#">setParent(parent)</a>	<a href="#">IChild</a>	Sets the parent object.

### Events details

#### parentchange

The parent object reference changed.

Data fields:

- `oldParent` - Old parent.
- `newParent` - New parent.

### Methods details

#### getParent

```
{Object} getParent()
```

**Returns** reference to the parent object.

**setParent**

```
{IChild} setParent(parent)
```

Sets the parent object.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<a href="#">parent</a> *	—	Type: Object Parent object.

\* Mandatory parameter/option.

**IChildOnMap**

Extends [IChild](#).

Interface for an object whose parent belongs to a particular map.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
IChildOnMap()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

**Events**

Name	Description
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .

**Methods**

Name	Returns	Description
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.

## Methods details

### getParent

```
{IParentOnMap|null} getParent()
```

**Returns** link to the parent object, or null if the parent element was not set.

### setParent

```
{IChildOnMap} setParent(parent)
```

Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>parent</code> *	—	Type: <code>IParentOnMap</code>  null Parent object.

\* Mandatory parameter/option.

## ICircleGeometry

Extends [ICircleGeometryAccess](#), [IGeometry](#).

Interface for the "Circle" geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
ICircleGeometry()
```

### Fields

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<code>options</code>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

### Events

Name	Description
<code>change</code>	Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>oldCoordinates</code> - Old coordinates of the center.</li> <li><code>newCoordinates</code> - New coordinates of the center.</li> <li><code>oldRadius</code> - Old radius.</li> <li><code>newRadius</code> - New radius.</li> </ul> Inherited from <a href="#">ICircleGeometryAccess</a> .

Name	Description
<a href="#">mapchange</a>	<p>Map reference changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> <p>Inherited from <a href="#">IGeometry</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">pixelgeometrychange</a>	<p>The pixel geometry changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>pixelGeometry - New <a href="#">IPixelGeometry</a> pixel geometry.</li> </ul> <p>Inherited from <a href="#">IGeometry</a>.</p>

## Methods

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	<p>Checks whether the passed point is located inside the circle.</p> <p>Inherited from <a href="#">ICircleGeometryAccess</a>.</p>
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	<p>Switches the object to "frozen" mode.</p> <p>Inherited from <a href="#">IFreezable</a>.</p>
<a href="#">getBounds()</a>	Number[][] null	<p>Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner.</p> <p>Inherited from <a href="#">IGeometry</a>.</p>
<a href="#">getClosest(anchorPosition)</a>	Object	<p>Searches for a point on the circle closest to the anchorPosition.</p> <p>Inherited from <a href="#">ICircleGeometryAccess</a>.</p>
<a href="#">getCoordinates()</a>	Number[] null	<p>Returns coordinates of the center of the circle.</p> <p>Inherited from <a href="#">ICircleGeometryAccess</a>.</p>
<a href="#">getMap()</a>	<a href="#">Map</a>  null	<p>Returns the current map.</p> <p>Inherited from <a href="#">IGeometry</a>.</p>

Name	Returns	Description
<a href="#">getPixelGeometry([options])</a>	<a href="#">IPixelGeometry</a>	Returns the pixel geometry corresponding to the given geometry, its options, and the map state.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getRadius()</a>	Number	Returns radius of the circle.  Inherited from <a href="#">ICircleGeometryAccess</a> .
<a href="#">getType()</a>	String	Returns the "Circle" string.
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false.  Inherited from <a href="#">IFreezable</a> .
<a href="#">setCoordinates(coordinates)</a>	<a href="#">ICircleGeometryAccess</a>	Sets the coordinates of the center of the circle.  Inherited from <a href="#">ICircleGeometryAccess</a> .
<a href="#">setMap(map)</a>		Sets the map.  Inherited from <a href="#">IGeometry</a> .
<a href="#">setRadius(radius)</a>	<a href="#">ICircleGeometryAccess</a>	Sets the radius of the circle.  Inherited from <a href="#">ICircleGeometryAccess</a> .
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode.  Inherited from <a href="#">IFreezable</a> .

## Methods details

### getType

```
{String} getType()
```

Returns the "Circle" string.

## ICircleGeometryAccess

Extends [IFreezable](#).

Interface for access to the "Circle" geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
ICircleGeometryAccess()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager for the object. Inherited from <a href="#">IFreezable</a> .

**Events**

Name	Description
<a href="#">change</a>	Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>oldCoordinates</code> - Old coordinates of the center.</li> <li><code>newCoordinates</code> - New coordinates of the center.</li> <li><code>oldRadius</code> - Old radius.</li> <li><code>newRadius</code> - New radius.</li> </ul>

**Methods**

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the circle.
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	Switches the object to "frozen" mode. Inherited from <a href="#">IFreezable</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for a point on the circle closest to the anchorPosition.
<a href="#">getCoordinates()</a>	Number[] null	Returns coordinates of the center of the circle.
<a href="#">getRadius()</a>	Number	Returns radius of the circle.
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from <a href="#">IFreezable</a> .
<a href="#">setCoordinates(coordinates)</a>	<a href="#">ICircleGeometryAccess</a>	Sets the coordinates of the center of the circle.
<a href="#">setRadius(radius)</a>	<a href="#">ICircleGeometryAccess</a>	Sets the radius of the circle.
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode. Inherited from <a href="#">IFreezable</a> .

**Events details****change**

Changed coordinates. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `oldCoordinates` - Old coordinates of the center.

- `newCoordinates` - New coordinates of the center.
- `oldRadius` - Old radius.
- `newRadius` - New radius.

### Methods details

#### contains

```
{Boolean} contains(position)
```

Checks whether the passed point is located inside the circle.

**Returns** an indicator for whether the point belongs to the circle.

#### Parameters:

Parameter	Default value	Description
<a href="#">position</a> *	—	Type: Number[] Coordinates of a point.

\* Mandatory parameter/option.

#### Example:

```
var myCircle = new ymaps.geometry.base.Circle([0, 0], 10);  
myCircle.contains([0, 10]); // => true
```

#### getClosest

```
{Object} getClosest(anchorPosition)
```

Searches for a point on the circle closest to the `anchorPosition`.

**Returns** an object with the following fields:

- `position` - Point on the circle closest to "anchorPosition".
- `distance` - Distance from "anchorPosition" to "position".

#### Parameters:

Parameter	Default value	Description
<a href="#">anchorPosition</a> *	—	Type: Number[] Coordinates of a point for which the nearest point on the circle is calculated.

\* Mandatory parameter/option.

#### Example:

```
var myCircle = new ymaps.geometry.base.Circle([0, 0], 10);  
myCircle.getClosest([0, 15]).position; // => [0, 10]
```

#### getCoordinates

```
{Number[]|null} getCoordinates()
```

**Returns** coordinates of the center of the circle.

**getRadius**

```
{Number} getRadius()
```

**Returns** radius of the circle.

**setCoordinates**

```
{ICircleGeometryAccess} setCoordinates(coordinates)
```

Sets the coordinates of the center of the circle.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>coordinates</code> *	—	Type: Number[] null  Coordinates of the center of the circle.

\* Mandatory parameter/option.

**setRadius**

```
{ICircleGeometryAccess} setRadius(radius)
```

Sets the radius of the circle.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>radius</code> *	—	Type: Number  Radius of the circle.

\* Mandatory parameter/option.

**ICollection**

Extends [IEventEmitter](#).

Interface for a collection.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
ICollection()
```

**Fields**

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

**Events**

Name	Description
<a href="#">add</a>	A child object was added.
<a href="#">remove</a>	A child object was deleted.

**Methods**

Name	Returns	Description
<a href="#">add(object)</a>	<a href="#">ICollection</a>	Adds a child object to the collection.
<a href="#">getIterator()</a>	<a href="#">IIterator</a>	Returns iterator for the collection.
<a href="#">remove(object)</a>	<a href="#">ICollection</a>	Removes a child object from the collection.

**Events details****add**

A child object was added.

**remove**

A child object was deleted.

**Methods details****add**

```
{ICollection} add(object)
```

Adds a child object to the collection.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<a href="#">object</a> *	—	Type: Object Object being added.

\* Mandatory parameter/option.

**getIterator**

```
{IIterator} getIterator()
```

**Returns** iterator for the collection.

**remove**

```
{ICollection} remove(object)
```

Removes a child object from the collection.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>object</code> *	—	Type: Object Object being removed.

\* Mandatory parameter/option.

## IContainerPane

Extends [IPane](#), [IPositioningContext](#).

Interface for the map pane intended for representations of the objects placed on the surface of the map. Allows you to transform global map pixels into a custom local coordinate system, thus enabling the object to be positioned within itself. Also informs about changes to the map status and thus allows the object to handle these changes.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IContainerPane()
```

### Fields

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

### Events

Name	Description
<code>actionbegin</code>	The start of pane movement. Instance of <a href="#">IEvent</a> .
<code>actionend</code>	The end of pane movement. Instance of <a href="#">IEvent</a> .
<code>clientpixelschange</code>	Change to the coordinate system of the client pixels.
<code>overflowchange</code>	Change to the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. Instance of <a href="#">IEvent</a> . Inherited from <a href="#">IPane</a> .
<code>viewportchange</code>	Change to a pane's viewport. Instance of <a href="#">IEvent</a> .
<code>zindexchange</code>	Change to the <code>zIndex</code> value of a pane. Instance of <a href="#">IEvent</a> . Inherited from <a href="#">IPane</a> .
<code>zoomchange</code>	Change to the current zoom level of the pane. Instance of <a href="#">IEvent</a> .

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destroys the pane. Inherited from <a href="#">IPane</a> .
<a href="#">fromClientPixels(clientPixelPoint)</a>	Number[]	Converts client pixel coordinates to global coordinates. Inherited from <a href="#">IPositioningContext</a> .
<a href="#">getElement()</a>	HTMLElement	Returns a reference to the pane's DOM container. Inherited from <a href="#">IPane</a> .
<a href="#">getMap()</a>	Map	Returns the map that the pane belongs to. Inherited from <a href="#">IPane</a> .
<a href="#">getOverflow()</a>	String	Returns value of the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. This parameter can take one of the following string values: <ul style="list-style-type: none"> <li>"visible" - When you go off the map container, the content of the pane remains visible.</li> <li>"hidden" - The viewport for the pane content is limited by the map container.</li> </ul> Inherited from <a href="#">IPane</a> .
<a href="#">getViewport()</a>	Number[][]	Returns the viewport for the pane, in client coordinates.
<a href="#">getZIndex()</a>	Number	Returns the zIndex of the pane. Inherited from <a href="#">IPane</a> .
<a href="#">getZoom()</a>	Number	Returns the current zoom level at which the positioning context works. Inherited from <a href="#">IPositioningContext</a> .
<a href="#">toClientPixels(globalPixelPoint)</a>	Number[]	Converts global pixel coordinates to client coordinates. Inherited from <a href="#">IPositioningContext</a> .

**Events details****actionbegin**

The start of pane movement. Instance of [IEvent](#).

**actionend**

The end of pane movement. Instance of [IEvent](#).

**clientpixelschange**

Change to the coordinate system of the client pixels.

**viewportchange**

Change to a pane's viewport. Instance of [IEvent](#).

**zoomchange**

Change to the current zoom level of the pane. Instance of [IEvent](#).

**Methods details****getViewport**

```
{Number[][]} getViewport()
```

**Returns** the viewport for the pane, in client coordinates.

**IControl**

Extends [IChildOnMap](#).

Control.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
IControl([options])
```

Interface for a control.

**Parameters:**

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object Control options.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager.

## Events

Name	Description
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getParent()</a>	<a href="#">IControlParent</a>  null	Returns link to the parent object, or null if the parent element was not set.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.

## Fields details

### options

```
{IOptionManager} options
```

Options manager.

### Methods details

#### getParent

```
{IControlParent|null} getParent()
```

**Returns** link to the parent object, or null if the parent element was not set.

#### setParent

```
{IChildOnMap} setParent(parent)
```

Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<a href="#">parent</a> *	—	Type: <a href="#">IControlParent</a>  null Parent object.

\* Mandatory parameter/option.

## IControlParent

Extends [IParentOnMap](#).

Interface of the parent object for the control.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IControlParent()
```

### Fields

Name	Type	Description
<a href="#">state</a>	<a href="#">IDataManager</a>	State manager.

### Events

Name	Description
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"><li>oldMap - Old map.</li><li>newMap - New map.</li></ul> Inherited from <a href="#">IParentOnMap</a> .

### Methods

Name	Returns	Description
<a href="#">getChildElement(child)</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the HTML element that should hold the child element.
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .

### Fields details

#### state

```
{IDataManager} state
```

State manager.

### Methods details

#### getChildElement

```
{vow.Promise} getChildElement(child)
```

**Returns** the promise object, which is confirmed by the HTML element that should hold the child element.

**Parameters:**

Parameter	Default value	Description
<code>child *</code>	—	Type: <a href="#">IControl</a> Child object.

\* Mandatory parameter/option.

## ICoordSystem

Interface for the map's coordinate system. This interface must be implemented if non-standard coordinates are used (such as cylindrical coordinates). To solve tasks with searching for the trajectory of movement on the earth's surface, use the [coordSystem.geo](#) object; on a Cartesian surface, use [coordSystem.cartesian](#).

See [coordSystem.geo](#) [coordSystem.cartesian](#)

[Constructor](#) | [Methods](#)

### Constructor

```
ICoordSystem()
```

### Methods

Name	Returns	Description
<a href="#">getDistance(point1, point2)</a>	Number	Returns the shortest distance (along a geodetic line) between the two set points (in meters).
<a href="#">solveDirectProblem(startPoint, direction, distance)</a>	Object	Solves the first (direct) <a href="#">geodesic problem</a> : where we will end up, if we start from a specified point and move in the specified direction for the specified distance, without turning. The following data is a solution for the direct geodetic problem: <ul style="list-style-type: none"> <li>• The end point.</li> <li>• The final direction.</li> <li>• The path function.</li> <li>• A function that allows to specify, for any given moment in time, which point we will be at and which direction we will be moving in.</li> </ul>

Name	Returns	Description
<code>solveInverseProblem(startPoint, endPoint[, reverseDirection])</code>	Object	Solves the second (inverse) <a href="#">geodetic problem</a> : construct the shortest route between two points on the mapped surface and determine the distance and direction of movement. Note that on the map of the Earth's surface, the shortest routes are shown as crooked lines. For geo objects in the API, you can enable the mode for displaying shortest distances between points using the "geodesic" option.

## Methods details

### getDistance

```
{Number} getDistance(point1, point2)
```

**Returns** the shortest distance (along a geodetic line) between the two set points (in meters).

#### Parameters:

Parameter	Default value	Description
<code>point1</code> *	—	Type: Number[] The first point.
<code>point2</code> *	—	Type: Number[] The second point.

\* Mandatory parameter/option.

#### Example:

```
// Calculating the distance between Moscow and New York.
// Coordinates of Moscow.
ymaps.geocode('Moscow').then(function (res) {
  var moscowCoords = res.geoObjects.get(0).geometry.getCoordinates();
  // Coordinates of New York.
  ymaps.geocode('New York').then(function (res) {
    var newYorkCoords = res.geoObjects.get(0).geometry.getCoordinates();
    // Distance.
    alert(ymaps.formatter.distance(
      ymaps.coordSystem.geo.getDistance(moscowCoords, newYorkCoords)
    ));
  });
});
```

### solveDirectProblem

```
{Object} solveDirectProblem(startPoint, direction, distance)
```

Solves the first (direct) [geodesic problem](#): where we will end up, if we start from a specified point and move in the specified direction for the specified distance, without turning. The following data is a solution for the direct geodetic problem:

- The end point.
- The final direction.
- The path function.

- A function that allows to specify, for any given moment in time, which point we will be at and which direction we will be moving in.

**Returns** object with following fields:

- `startPoint` - The starting point in geocoordinates.
- `startDirection` - The starting direction of movement.
- `endPoint` - The end point in geocoordinates.
- `endDirection` - The final direction of movement.
- `distance` - The distance in meters.
- `pathFunction` - A function that accepts a number from 0 to 1 (the portion of the path completed) and returns a structure with the fields "point" and "direction".

**Parameters:**

Parameter	Default value	Description
<code>startPoint</code> *	—	Type: Number[]  Point of departure.
<code>direction</code> *	—	Type: Number[]  Direction. Set as a vector (an increment of coordinates) - either [dlat, dlon] or [dlon, dlat], depending on the "coordorder" parameter. In order to get the azimuth from a direction specified like this (the azimuth is the angle between the direction of movement and North), we need to calculate the arctangent of the dlon/dlat amount (in JavaScript - this is a standard function <code>Math.atan2(dlon, dlat)</code> ). In order to calculate the direction of movement from the azimuth "a", put <code>dlat = cos(a)</code> , <code>dlon = sin(a)</code> .
<code>distance</code> *	—	Type: Number  The distance walked, in meters.

\* Mandatory parameter/option.

**Example:**

```
// Let's assume that we took off from Domodedovo airport (Moscow) going
// northeast and flew straight for 200 kilometers. We'll use placemarks to show our path
// on the map.

// Finding the coordinates of the starting point, using geocoding.
ymaps.geocode('Domodedovo airport').then(function (res) {
  var startPoint = res.geoObjects.get(0).geometry.getCoordinates();
  // Moving northeast, azimuth of 45 degrees
  // or radian pi/4.
  var azimuth = Math.PI / 4;
  // Direction of movement.
  var direction = [Math.cos(azimuth), Math.sin(azimuth)];
  // Path function.
  var path = ymaps.coordSystem.geo.solveDirectProblem(startPoint, direction, 2e5).pathFunction;

  // Showing the path on the map using placemarks set every 10 km.
  for (var i = 0; i <= 20; i++) {
    map.geoObjects.add(new ymaps.Placemark(path(i/20).point));
  }
});
```

```
});
```

## solveInverseProblem

```
{Object} solveInverseProblem(startPoint, endPoint[, reverseDirection])
```

Solves the second (inverse) [geodetic problem](#): construct the shortest route between two points on the mapped surface and determine the distance and direction of movement. Note that on the map of the Earth's surface, the shortest routes are shown as crooked lines. For geo objects in the API, you can enable the mode for displaying shortest distances between points using the "geodesic" option.

**Returns** object with following fields:

- `startPoint` - The starting point in geocoordinates.
- `startDirection` - The starting direction of movement.
- `endPoint` - The end point in geocoordinates.
- `endDirection` - The final direction of movement.
- `distance` - The distance in meters.
- `pathFunction` - A path function that accepts a number from 0 to 1 (the portion of the path completed) and returns a structure with the fields "point" and "direction".

**Parameters:**

Parameter	Default value	Description
<code>startPoint</code> *	—	Type: Number[] Point of departure.
<code>endPoint</code> *	—	Type: Number[] Point of arrival.
<code>reverseDirection</code>	false	Type: Boolean Direction of movement. "false" - select the shortest arc; "true" - select the opposite of the shortest arc.

\* Mandatory parameter/option.

**Example:**

```
// Constructing the shortest route from Kaliningrad to Vladivostok.
// Finding the coordinates of Kaliningrad.
ymaps.geocode('Kaliningrad').then(function (res) {
  var startPoint = res.geoObjects.get(0).geometry.getCoordinates();
  // Finding the coordinates of Vladivostok.
  ymaps.geocode('Vladivostok').then(function (res) {
    var endPoint = res.geoObjects.get(0).geometry.getCoordinates();
    // Finding the function of the path between two points.
    var path = ymaps.coordSystem.geo.solveInverseProblem(startPoint, endPoint).pathFunction;
    // Showing the path with 20 points.
    for (var i = 0; i <= 20; i++) {
      // Finding an intermediate point.
      var position = path(i/20).point;
      // Adding a placemark to an intermediate point.
      map.geoObjects.add(new ymaps.Placemark(position, {
        // Showing the distance covered in the placemark content.
        iconContent: ymaps.formatter.distance(
          ymaps.coordSystem.geo.getDistance(startPoint, position)
        )
      }, {
        preset: 'isladns#redStretchyIcon'
      }));
    }
  });
});
```

## ICopyrightsAccessor

Extends [ICopyrightsProvider](#).

Interface for an object that provides access to user copyright information that was added to the map using the method `map.Copyrights.add`.

**See** `map.Copyrights.add`

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
ICopyrightsAccessor()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

### Events

Name	Description
<a href="#">copyrightschange</a>	Event for changes to copyright information. Inherited from <a href="#">ICopyrightsProvider</a> .

### Methods

Name	Returns	Description
<a href="#">getCopyrights(coords, zoom)</a>	<a href="#">vow.Promise</a>	Requests information about copyrights at the specified point with the specified zoom.  Inherited from <a href="#">ICopyrightsProvider</a> .
<a href="#">remove()</a>		Removes copyright information that was added via this object.  Inherited from <a href="#">ICopyrightsProvider</a> .
<a href="#">setCopyrights(copyrights)</a>		Sets a new value for the copyright information that was added via this object.  Inherited from <a href="#">ICopyrightsProvider</a> .

## ICopyrightsProvider

Extends [IEventEmitter](#).

Copyright information provider.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
ICopyrightsProvider()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

**Events**

Name	Description
<a href="#">copyrightschange</a>	Event for changes to copyright information.

**Methods**

Name	Returns	Description
<a href="#">getCopyrights(coords, zoom)</a>	<a href="#">vow.Promise</a>	Requests information about copyrights at the specified point with the specified zoom.
<a href="#">remove()</a>		Removes copyright information that was added via this object.
<a href="#">setCopyrights(copyrights)</a>		Sets a new value for the copyright information that was added via this object.

**Events details****copyrightschange**

Event for changes to copyright information.

**Methods details****getCopyrights**

```
{vow.Promise} getCopyrights(coords, zoom)
```

Requests information about copyrights at the specified point with the specified zoom.

**Returns** Promise that will be resolved and will pass as a result an array of strings or DOM elements with information about copyrights.

**Parameters:**

Parameter	Default value	Description
<a href="#">coords</a> *	—	Type: <code>Number[]</code>  The point on the map that copyright information is being requested for.

Parameter	Default value	Description
<a href="#">zoom</a> *	—	Type: Number  The zoom level that copyright information is being requested for.

\* Mandatory parameter/option.

### remove

```
{ } remove ()
```

Removes copyright information that was added via this object.

### setCopyrights

```
{ } setCopyrights (copyrights)
```

Sets a new value for the copyright information that was added via this object.

#### Parameters:

Parameter	Default value	Description
<a href="#">copyrights</a> *	—	Type: String HTMLElement String[] HTMLElement[]  Copyright information.

\* Mandatory parameter/option.

## ICustomizable

Extends [IEventEmitter](#).

Interface of an object that can be configured using the global options register.

[Constructor](#) | [Fields](#) | [Events](#)

### Constructor

```
ICustomizable ()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionsManager</a>	Options manager.

### Events

Name	Description
<a href="#">optionschange</a>	Change to the object options.

**Fields details****options**

```
{IOptionManager} options
```

Options manager.

**Events details****optionschange**

Change to the object options.

**IDataManager**

Extends [IEventEmitter](#).

Interface for the custom data manager.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
IDataManager ()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

**Events**

Name	Description
<a href="#">change</a>	Data change.

**Methods**

Name	Returns	Description
<a href="#">get(path, defaultValue)</a>	Object	Returns value of data fields.

**Events details****change**

Data change.

**Methods details****get**

```
{Object} get (path, defaultValue)
```

**Returns** value of data fields.

**Parameters:**

Parameter	Default value	Description
<code>path</code> *	—	Type: String  String with the property name; can contain '.'.
<code>defaultValue</code> *	—	Type: Object  Default value.

\* Mandatory parameter/option.

#### Example:

```
dataManager.get("icon.shadow.size", [10, 10]);
```

## IDomEvent

Extends [IEvent](#).

The DOM event object in the Yandex.Maps API system. Wraps the browser's source DOM event in order to normalize the data field names. This means you can use the "get" and "callMethod" methods to access the fields and methods of the source DOM event. In addition, standardization occurs automatically for those fields and methods that are implemented differently in different browsers. So `event.callMethod('stopPropagation')` stops propagation of the DOM event in all browsers, including Internet Explorer.

[Constructor](#) | [Methods](#)

### Constructor

```
IDomEvent(originalEvent)
```

#### Parameters:

Parameter	Default value	Description
<code>originalEvent</code> *	—	Type: Object  Source DOM event.

\* Mandatory parameter/option.

### Methods

Name	Returns	Description
<code>allowMapEvent()</code>		Allows the propagation of the event to the map.  Inherited from <a href="#">IEvent</a> .
<code>callMethod(name)</code>		Calls the specified method from the source event. The second and following arguments are passed to the method with the call.  Inherited from <a href="#">IEvent</a> .

Name	Returns	Description
<a href="#">get(name)</a>	Object	Returns the specified event property. Using this method, you can get access both to the properties of the source event and to additional properties provided by the Yandex.Maps API.
<a href="#">getSourceEvent()</a>	<a href="#">IDomEvent</a>	Returns source DOM event.
<a href="#">isDefaultPrevented()</a>	Boolean	Returns true if the default reaction to the event has been canceled.  Inherited from <a href="#">IEvent</a> .
<a href="#">isImmediatePropagationStopped()</a>	Boolean	Returns true if the event processing has been interrupted.  Inherited from <a href="#">IEvent</a> .
<a href="#">isMapEventAllowed()</a>	Boolean	Returns true if the map event is enabled.  Inherited from <a href="#">IEvent</a> .
<a href="#">isPropagationStopped()</a>	Boolean	Returns true if event propagation has been interrupted.  Inherited from <a href="#">IEvent</a> .
<a href="#">preventDefault()</a>		Cancels the default reaction to an event within the Yandex.Maps API event system. Calling this method does not affect how the browser processes the default action for the source DOM event.
<a href="#">stopImmediatePropagation()</a>		Stops event processing in the Yandex.Maps API event system. I.e. after calling this method, no handler for this event will be called. Calling this method does not affect the processing of the original DOM-event at the browser level.
<a href="#">stopPropagation()</a>		Stops propagation of the DOM event in the Yandex.Maps API event system. Calling this method does not affect propagation of the source DOM event through the DOM tree.

## Methods details

### get

```
{Object} get (name)
```

Returns the specified event property. Using this method, you can get access both to the properties of the source event and to additional properties provided by the Yandex.Maps API.

**Returns** property value.

#### Parameters:

Parameter	Default value	Description
<code>name</code> *	—	Type: String  Property name. Additional properties are supported: <ul style="list-style-type: none"><li>'propagatedData' - Event data that is saved during event propagation through the DOM tree.</li><li>'position' - An optional field that contains the coordinates of the event, relative to the document.</li></ul>

\* Mandatory parameter/option.

### getSourceEvent

```
{IDomEvent} getSourceEvent()
```

**Returns** source DOM event.

### preventDefault

```
{ } preventDefault()
```

Cancels the default reaction to an event within the Yandex.Maps API event system. Calling this method does not affect how the browser processes the default action for the source DOM event.

### stopImmediatePropagation

```
{ } stopImmediatePropagation()
```

Stops event processing in the Yandex.Maps API event system. I.e. after calling this method, no handler for this event will be called. Calling this method does not affect the processing of the original DOM-event at the browser level.

### stopPropagation

```
{ } stopPropagation()
```

Stops propagation of the DOM event in the Yandex.Maps API event system. Calling this method does not affect propagation of the source DOM event through the DOM tree.

## IDomEventEmitter

Extends [IEventEmitter](#).

Interface of an object that generates a "DOM event". This interface declares a list of events that are used for displaying interactive user manipulation of various objects in the Yandex.Maps API environment. Objects implementing this interface are not required to actually generate all the events declared by the interface.

[Constructor](#) | [Fields](#) | [Events](#)

### Constructor

```
IDomEventEmitter()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.

### Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .
<a href="#">mouseleave</a>	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .
<a href="#">mousemove</a>	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .
<a href="#">mouseup</a>	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .
<a href="#">multitouchend</a>	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.

Name	Description
<a href="#">multitouchmove</a>	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: <ul style="list-style-type: none"> <li>clientX - X coordinate of the touch relative to the viewable area of the browser.</li> <li>clientY - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>pageX - X coordinate of the touch relative to the beginning of the document.</li> <li>pageY - Y coordinate of the touch relative to the beginning of the document.</li> </ul>
<a href="#">multitouchstart</a>	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: <ul style="list-style-type: none"> <li>clientX - X coordinate of the touch relative to the viewable area of the browser.</li> <li>clientY - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>pageX - X coordinate of the touch relative to the beginning of the document.</li> <li>pageY - Y coordinate of the touch relative to the beginning of the document.</li> </ul>
<a href="#">wheel</a>	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .

## Fields details

### events

```
{EventManager} events
```

Event manager.

### Events details

#### click

Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#).

#### contextmenu

Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#).

#### dblclick

Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#).

#### mousedown

Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#).

**mouseenter**

Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#).

**mouseleave**

Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#).

**mousemove**

Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#).

**mouseup**

Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#).

**multitouchend**

End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface.

**multitouchmove**

Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields:

- clientX - X coordinate of the touch relative to the viewable area of the browser.
- clientY - Y coordinate of the touch relative to the viewable area of the browser.
- pageX - X coordinate of the touch relative to the beginning of the document.
- pageY - Y coordinate of the touch relative to the beginning of the document.

**multitouchstart**

Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the IMultiTouchEvent interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields:

- clientX - X coordinate of the touch relative to the viewable area of the browser.
- clientY - Y coordinate of the touch relative to the viewable area of the browser.
- pageX - X coordinate of the touch relative to the beginning of the document.
- pageY - Y coordinate of the touch relative to the beginning of the document.

**wheel**

Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#).

**IDomTile**

Extends [ITile](#).

Interface for tiles that make up a DOM object.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
IDomTile(url)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">url</a> *	—	Type: String Tile URL.

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

**Events**

Name	Description
<a href="#">ready</a>	Tile ready event. Inherited from <a href="#">ITile</a> .

**Methods**

Name	Returns	Description
<a href="#">destroy()</a>		Destroys the tile. Inherited from <a href="#">ITile</a> .
<a href="#">isReady()</a>	Boolean	Checks tile readiness. Inherited from <a href="#">ITile</a> .
<a href="#">renderAt(context, clientBounds, animate)</a>		Adds a tile to the parent HTML element.

**Methods details****renderAt**

```
{ } renderAt(context, clientBounds, animate)
```

Adds a tile to the parent HTML element.

**Parameters:**

Parameter	Default value	Description
<a href="#">context</a> *	—	Type: HTMLElement Parent HTML element

Parameter	Default value	Description
<a href="#">clientBounds</a> *	—	Type: Number[][]  The area in client coordinates that the tile should occupy.
<a href="#">animate</a> *	—	Type: Boolean  If true, rendering is animated; if false, it is not.

\* Mandatory parameter/option.

## IEvent

Event that is fired by the IEventManager event manager.

[Constructor](#) | [Methods](#)

### Constructor

```
IEvent()
```

### Methods

Name	Returns	Description
<a href="#">allowMapEvent()</a>		Allows the propagation of the event to the map.
<a href="#">callMethod(name)</a>		Calls the specified method from the source event. The second and following arguments are passed to the method with the call.
<a href="#">get(name)</a>	Object	Returns an event property for a key. Using this method, you can get access both to the properties of the source event and to additional properties provided by the Yandex.Maps API.
<a href="#">getSourceEvent()</a>	IEvent null	Returns source event.
<a href="#">isDefaultPrevented()</a>	Boolean	Returns true if the default reaction to the event has been canceled.
<a href="#">isImmediatePropagationStopped()</a>	Boolean	Returns true if the event processing has been interrupted.
<a href="#">isMapEventAllowed()</a>	Boolean	Returns true if the map event is enabled.
<a href="#">isPropagationStopped()</a>	Boolean	Returns true if event propagation has been interrupted.

Name	Returns	Description
<a href="#">preventDefault()</a>		Cancels the default reaction to an event within the Yandex.Maps API event system.
<a href="#">stopImmediatePropagation()</a>		Stops event processing in the Yandex.Maps API event system. I.e. after calling this method, no handler for this event will be called.
<a href="#">stopPropagation()</a>		Stops event propagation in the Yandex.Maps API event system.

## Methods details

### allowMapEvent

```
{ } allowMapEvent()
```

Allows the propagation of the event to the map.

### callMethod

```
{ } callMethod(name)
```

Calls the specified method from the source event. The second and following arguments are passed to the method with the call.

#### Parameters:

Parameter	Default value	Description
<a href="#">name</a> *	—	Type: String Method name.

\* Mandatory parameter/option.

### get

```
{Object} get(name)
```

Returns an event property for a key. Using this method, you can get access both to the properties of the source event and to additional properties provided by the Yandex.Maps API.

**Returns** property value.

#### Parameters:

Parameter	Default value	Description
<a href="#">name</a> *	—	Type: String Property name.

\* Mandatory parameter/option.

**getSourceEvent**

```
{IEvent|null} getSourceEvent()
```

**Returns** source event.

**isDefaultPrevented**

```
{Boolean} isDefaultPrevented()
```

**Returns** true if the default reaction to the event has been canceled.

**isImmediatePropagationStopped**

```
{Boolean} isImmediatePropagationStopped()
```

**Returns** true if the event processing has been interrupted.

**isMapEventAllowed**

```
{Boolean} isMapEventAllowed()
```

**Returns** true if the map event is enabled.

**isPropagationStopped**

```
{Boolean} isPropagationStopped()
```

**Returns** true if event propagation has been interrupted.

**preventDefault**

```
{ } preventDefault()
```

Cancels the default reaction to an event within the Yandex.Maps API event system.

**stopImmediatePropagation**

```
{ } stopImmediatePropagation()
```

Stops event processing in the Yandex.Maps API event system. I.e. after calling this method, no handler for this event will be called.

**stopPropagation**

```
{ } stopPropagation()
```

Stops event propagation in the Yandex.Maps API event system.

**IEventController**

Interface for an event controller. For controlling how events are subscribed to and unsubscribed from on a particular event manager.

[Constructor](#) | [Methods](#)

**Constructor**

```
IEventController()
```

**Methods**

Name	Description
<a href="#">onStartListening(events, type)</a>	Called on the first subscription to the specified event type using the specified event manager. This method is optional.
<a href="#">onStopListening(events, type)</a>	Called when a particular event type stops listening on the specified event manager (the last subscription is deleted). This method is optional.

**Methods details****onStartListening**

```
{ } onStartListening(events, type)
```

Called on the first subscription to the specified event type using the specified event manager. This method is optional.

**Parameters:**

Parameter	Default value	Description
<a href="#">events</a> *	—	Type: <a href="#">IEventManager</a> Event manager.
<a href="#">type</a> *	—	Type: String Type of event.

\* Mandatory parameter/option.

**onStopListening**

```
{ } onStopListening(events, type)
```

Called when a particular event type stops listening on the specified event manager (the last subscription is deleted). This method is optional.

**Parameters:**

Parameter	Default value	Description
<a href="#">events</a> *	—	Type: <a href="#">IEventManager</a> Event manager.
<a href="#">type</a> *	—	Type: String Type of event.

\* Mandatory parameter/option.

**IEventEmitter**

Interface of the object that events can be listened on.

[Constructor](#) | [Fields](#)

**Constructor**

```
IEmitter()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.

**Fields details****events**

```
{IEventManager} events
```

Event manager.

**IEventGroup**

A group of event listeners.

[Constructor](#) | [Methods](#)

**Constructor**

```
IEventGroup()
```

**Methods**

Name	Returns	Description
<a href="#">add(types, callback[, context[, priority]])</a>	<a href="#">IEventGroup</a>	Adds an event listener.
<a href="#">remove(types, callback[, context[, priority]])</a>	<a href="#">IEventGroup</a>	Deletes an event listener from the group.
<a href="#">removeAll()</a>	<a href="#">IEventGroup</a>	Deletes all event listeners from the group.

**Methods details****add**

```
{IEventGroup} add(types, callback[, context[, priority]])
```

Adds an event listener.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<a href="#">types</a> *	—	Type: <code>String String[]</code>  Type or array of types for the event.

Parameter	Default value	Description
<code>callback *</code>	—	Type: Function  Handler function. The event object is passed to the function as a parameter.
<code>context</code>	—	Type: Object  Context for the handler function.
<code>priority</code>	0	Type: Integer  Subscription priority.

\* Mandatory parameter/option.

### remove

```
{IEventGroup} remove(types, callback[, context[, priority]])
```

Deletes an event listener from the group.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>types *</code>	—	Type: String String[]  Type or array of types for the events.
<code>callback *</code>	—	Type: Function  Handler function. The event object is passed to the function as a parameter.
<code>context</code>	—	Type: Object  Context for the handler function.
<code>priority</code>	0	Type: Integer  Subscription priority.

\* Mandatory parameter/option.

### removeAll

```
{IEventGroup} removeAll()
```

Deletes all event listeners from the group.

**Returns** self-reference.

## IEventManager

Extends [IEventTrigger](#).

Event manager. Using an event manager, you can subscribe to and unsubscribe from events, as well as initiate the events themselves.

[Constructor](#) | [Methods](#)

## Constructor

```
EventManager()
```

## Methods

Name	Returns	Description
<a href="#">add</a> (types, callback[, context[, priority]])	<a href="#">EventManager</a>	Adds a new subscription.
<a href="#">fire</a> (type[, event])	<a href="#">EventManager</a>	Triggers an event.
<a href="#">getParent</a> ()	<a href="#">EventManager</a>  null	Returns reference to the parent event manager.
<a href="#">group</a> ()	<a href="#">EventGroup</a>	Returns the group of event listeners associated with the given event manager.
<a href="#">once</a> (types, callback[, context[, priority]])	<a href="#">EventManager</a>	Adds a listener, which calls the handler function only one time.
<a href="#">remove</a> (types, callback[, context[, priority]])	<a href="#">EventManager</a>	Removes an existing subscription.
<a href="#">setParent</a> (parent)		Sets the parent event manager.

## Methods details

### add

```
{EventManager} add(types, callback[, context[, priority]])
```

Adds a new subscription.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<a href="#">types</a> *	—	Type: String String[]  Type or array of types for the event.
<a href="#">callback</a> *	—	Type: Function  Handler function for the event. An object describing the event is passed to the function as a parameter. It can be either an arbitrary object, or implement the interface <a href="#">IEvent</a> .

Parameter	Default value	Description
<code>context</code>	—	Type: Object Context for the handler.
<code>priority</code>	0	Type: Integer Subscription priority.

\* Mandatory parameter/option.

### fire

```
{EventManager} fire(type[, event])
```

Triggers an event.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>type</code> *	—	Type: String Type of event.
<code>event</code>	—	Type: Object Event Event. If a hash with data is passed, the <code>createEventObject</code> method will be called for it and further actions will be performed with the newly created event.

\* Mandatory parameter/option.

### getParent

```
{EventManager|null} getParent()
```

**Returns** reference to the parent event manager.

### group

```
{EventManager} group()
```

**Returns** the group of event listeners associated with the given event manager.

### once

```
{EventManager} once(types, callback[, context[, priority]])
```

Adds a listener, which calls the handler function only one time.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>types *</code>	—	Type: String String[]  Type or array of types for the event.
<code>callback *</code>	—	Type: Function  Handler function for the event. The function is passed an object of the event <a href="#">IEvent</a> .
<code>context</code>	—	Type: Object  Context for the handler.
<code>priority</code>	0	Type: Integer  Subscription priority.

\* Mandatory parameter/option.

### remove

```
{IEventManager} remove(types, callback[, context[, priority]])
```

Removes an existing subscription.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>types *</code>	—	Type: String String[]  Type or array of types for the event.
<code>callback *</code>	—	Type: Function  Handler function for the event. The function is passed an object of the event <a href="#">IEvent</a> .
<code>context</code>	—	Type: Object  Context for the handler.
<code>priority</code>	0	Type: Integer  Subscription priority.

\* Mandatory parameter/option.

### setParent

```
{ } setParent(parent)
```

Sets the parent event manager.

#### Parameters:

Parameter	Default value	Description
<a href="#">parent</a> *	—	Type: <a href="#">IEventManager</a>  null Parent event manager.

\* Mandatory parameter/option.

## IEventPane

Extends [IDomEventEmitter](#), [IPane](#).

Interface for a map events pane. Allows to trace the DOM events on the map but, unlike [IContainerPane](#), does not allow to place representations of map objects inside its DOM element.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IEventPane ()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

### Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">overflowchange</a>	<p>Change to the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. Instance of <a href="#">IEvent</a>.</p> <p>Inherited from <a href="#">IPane</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">zindexchange</a>	<p>Change to the <code>zIndex</code> value of a pane. Instance of <a href="#">IEvent</a>.</p> <p>Inherited from <a href="#">IPane</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">destroy()</a>		Destroys the pane. Inherited from <a href="#">IPane</a> .
<a href="#">getElement()</a>	HTMLElement	Returns a reference to the pane's DOM container. Inherited from <a href="#">IPane</a> .
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns the map that the pane belongs to. Inherited from <a href="#">IPane</a> .
<a href="#">getOverflow()</a>	String	Returns value of the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. This parameter can take one of the following string values: <ul style="list-style-type: none"> <li>"visible" - When you go off the map container, the content of the pane remains visible.</li> <li>"hidden" - The viewport for the pane content is limited by the map container.</li> </ul> Inherited from <a href="#">IPane</a> .
<a href="#">getZIndex()</a>	Number	Returns the zIndex of the pane. Inherited from <a href="#">IPane</a> .

**IEventTrigger**

The object which triggers the events.

[Constructor](#) | [Methods](#)

**Constructor**

```
IEventTrigger()
```

**Methods**

Name	Returns	Description
<a href="#">fire(type[, eventObject])</a>	<a href="#">IEventTrigger</a>	Triggers an event.

**Methods details****fire**

```
{IEventTrigger} fire(type[, eventObject])
```

Triggers an event.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>type</code> *	—	Type: String  Type of event.
<code>eventObject</code>	—	Type: Object  <a href="#">IEvent</a>  Object that describes the event. It can be either an arbitrary object, or implement the interface <a href="#">IEvent</a> . In the latter case, after each handler is called, the value of the <code>isImmediatePropagationStopped()</code> method is checked, and if true, event propagation is stopped immediately.

\* Mandatory parameter/option.

## IEventWorkflowController

Extends [IEventController](#).

Interface for an event controller that can affect event propagation through the tree.

[Constructor](#) | [Methods](#)

### Constructor

```
IEventWorkflowController()
```

### Methods

Name	Description
<code>onAfterEventFiring(events, type[, event])</code>	Function that is called after an event has been handled by the event manager. This method is optional.
<code>onBeforeEventFiring(events, type[, event])</code>	Function that is called before an event is handled by the event manager. This method is optional.
<code>onStartListening(events, type)</code>	Called on the first subscription to the specified event type using the specified event manager. This method is optional.  Inherited from <a href="#">IEventController</a> .
<code>onStopListening(events, type)</code>	Called when a particular event type stops listening on the specified event manager (the last subscription is deleted). This method is optional.  Inherited from <a href="#">IEventController</a> .

### Methods details

#### onAfterEventFiring

```
{ } onAfterEventFiring(events, type[, event])
```

Function that is called after an event has been handled by the event manager. This method is optional.

#### Parameters:

Parameter	Default value	Description
<code>events</code> *	—	Type: <a href="#">IEventManager</a> Event manager.
<code>type</code> *	—	Type: String Type of event.
<code>event</code>	—	Type: <a href="#">IEvent</a> Event.

\* Mandatory parameter/option.

#### onBeforeEventFiring

```
{ } onBeforeEventFiring(events, type[, event])
```

Function that is called before an event is handled by the event manager. This method is optional.

#### Parameters:

Parameter	Default value	Description
<code>events</code> *	—	Type: <a href="#">IEventManager</a> Event manager.
<code>type</code> *	—	Type: String Type of event.
<code>event</code>	—	Type: <a href="#">IEvent</a> Event.

\* Mandatory parameter/option.

## IExpandableControlLayout

Extends [ILayout](#).

Interface for a layout that can be in the collapsed or expanded state.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
IExpandableControlLayout()
```

#### Fields

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .

## Events

Name	Description
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">collapse</a>	<p>Event that initiates collapsing an object.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">emptinesschange</a>	<p>Change to the empty layout indicator. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">expand</a>	<p>Event that initiates expanding an object.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">parentelementchange</a>	<p>Change to the parent element. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the layout. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		<p>Destructor. Called when activity with the layout is finished.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">getData()</a>	Object	<p>Returns layout data object.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">getParentElement()</a>	HTMLElement	<p>Returns parent HTML element.</p> <p>Inherited from <a href="#">ILayout</a>.</p>

Name	Returns	Description
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element.  Inherited from <a href="#">ILayout</a> .
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.  Inherited from <a href="#">ILayout</a> .
<a href="#">setData(data)</a>		Sets layout data.  Inherited from <a href="#">ILayout</a> .
<a href="#">setParentElement(parent)</a>		Adds the layout to the DOM tree.  Inherited from <a href="#">ILayout</a> .

### Events details

#### collapse

Event that initiates collapsing an object.

#### expand

Event that initiates expanding an object.

## IFreezable

Interface for an object with a state change event that can be disabled. Object that implements [IFreezable](#) it can function in one of the following modes:

- 1. Active. In this mode, each change to the internal state of the object generates an [IFreezable.event:change](#) event.
- 2. Frozen. In this mode, changes to the object state do not cause an [IFreezable.event:change](#) event, but if changes occurred, the [IFreezable.event:change](#) event will be generated once when switching to active mode.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IFreezable()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager for the object.

## Events

Name	Description
<a href="#">change</a>	Change to the internal state of the object.

## Methods

Name	Returns	Description
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	Switches the object to "frozen" mode.
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false.
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode.

## Fields details

### events

```
{EventManager} events
```

Event manager for the object.

### Events details

#### change

Change to the internal state of the object.

### Methods details

#### freeze

```
{IFreezable} freeze()
```

Switches the object to "frozen" mode.

**Returns** self-reference.

#### isFrozen

```
{Boolean} isFrozen()
```

**Returns** true if the object is in "frozen" mode, otherwise false.

#### unfreeze

```
{IFreezable} unfreeze()
```

Switches the object to active mode.

**Returns** self-reference.

## IGeocodeProvider

Interface for a geocoder provider.

[Constructor](#) | [Methods](#)**Constructor**`IGeocodeProvider()`**Methods**

Name	Returns	Description
<code>geocode(request[, options])</code>	<code>vow.Promise</code>	Sends a request for geocoding. A handler function for processing geocoding results can be added via the returned promise object. The object input to the handler function can contain only the following types of fields: <code>geoObjects</code> , <code>layers</code> , <code>mapState</code> , and <code>metaData</code> .
<code>suggest(request[, options])</code>	<code>vow.Promise</code>	Sends a request for search suggestions. Returns a promise object that is either rejected with an error, or confirmed by an array of objects in the format <code>{ displayName: "Mitishi, Moscow region", value: "Russia, Moscow region, Mitishi " }</code> . The <code>displayName</code> field represents the toponym in a user-friendly way, and the <code>value</code> field represents the value which should be inserted into the search field after the user selects the suggestion.  This method is optional.

**Methods details****geocode**`{vow.Promise} geocode(request[, options])`

Sends a request for geocoding. A handler function for processing geocoding results can be added via the returned promise object. The object input to the handler function can contain only the following types of fields: `geoObjects`, `layers`, `mapState`, and `metaData`.

**Returns** Promise object.

**Parameters:**

Parameter	Default value	Description
<code>request</code> *	—	Type: String  Request string.

Parameter	Default value	Description
<code>options</code>	—	Type: Object  Options.
<code>options.boundedBy</code>	—	Type: Number[][]  A rectangular area on the map, where the object being searched for is presumably located. Must be set as an array, such as <code>[[30, 40], [50, 50]]</code> .
<code>options.results</code>	—	Type: Number  Maximum number of results to be returned.
<code>options.skip</code>	—	Type: Number  Number of results that must be skipped.
<code>options.strictBounds</code>	—	Type: Boolean  Search only inside the area defined by the "boundedBy" option.

\* Mandatory parameter/option.

### suggest

```
{vow.Promise} suggest(request[, options])
```

Sends a request for search suggestions. Returns a promise object that is either rejected with an error, or confirmed by an array of objects in the format `{ displayName: "Mitishi, Moscow region", value: "Russia, Moscow region, Mitishi " }`. The `displayName` field represents the toponym in a user-friendly way, and the `value` field represents the value which should be inserted into the search field after the user selects the suggestion.

This method is optional.

**Returns** a Promise object.

#### Parameters:

Parameter	Default value	Description
<code>request</code> *	—	Type: String  Request string.
<code>options</code>	—	Type: Object  Options.

Parameter	Default value	Description
<a href="#">options.boundedBy</a>	—	Type: Number[][]  A rectangular area on the map, where the object being searched for is presumably located. Must be set as an array, such as <code>[[30, 40], [50, 50]]</code> .
<a href="#">options.results</a>	—	Type: Number  Maximum number of results to be returned.
<a href="#">options.strictBounds</a>	—	Type: Boolean  Search only inside the area defined by the "boundedBy" option.

\* Mandatory parameter/option.

## IGeometry

Extends [IBaseGeometry](#), [ICustomizable](#).

Interface of a geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IGeometry()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

### Events

Name	Description
<a href="#">mapchange</a>	Map reference changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>oldMap</code> - Old map.</li> <li><code>newMap</code> - New map.</li> </ul>
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .

Name	Description
<a href="#">pixelgeometrychange</a>	<p>The pixel geometry changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>pixelGeometry</code> - New <a href="#">IPixelGeometry</a> pixel geometry.</li> </ul>

## Methods

Name	Returns	Description
<a href="#">getBounds()</a>	<code>Number[][] null</code>	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner.
<a href="#">getMap()</a>	<code>Map null</code>	Returns the current map.
<a href="#">getPixelGeometry([options])</a>	<a href="#">IPixelGeometry</a>	Returns the pixel geometry corresponding to the given geometry, its options, and the map state.
<a href="#">getType()</a>	<code>String</code>	Returns ID of the geometry type.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">setMap(map)</a>		Sets the map.

## Events details

### mapchange

Map reference changed. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `oldMap` - Old map.
- `newMap` - New map.

### pixelgeometrychange

The pixel geometry changed. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `pixelGeometry` - New [IPixelGeometry](#) pixel geometry.

## Methods details

### getBounds

```
{Number[][]|null} getBounds()
```

**Returns** coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner.

### Example:

```
// Setting the map center and zoom so that the geometry is displayed in its entirety inside the visible area.
map.setBounds(myGeometry.getBounds());
```

## getMap

```
{Map|null} getMap()
```

Returns the current map.

## getPixelGeometry

```
{IPixelGeometry} getPixelGeometry([options])
```

Returns the pixel geometry corresponding to the given geometry, its options, and the map state.

### Parameters:

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object  Hash of options that allow to exclude part of the current geometry options for this calculation.

### Example:

```
// Getting a pixel representation of the geometry with geodesics calculated, but without optimizing deletion of
invisible points.
myGeometry.getPixelGeometry({
  geodesic: true,
  simplification: false
}).getCoordinates();
```

## setMap

```
{ } setMap (map)
```

Sets the map.

### Parameters:

Parameter	Default value	Description
<a href="#">map</a> *	—	Type: <a href="#">Map</a>  null  Reference to the map.

\* Mandatory parameter/option.

## IGeometryEditor

Extends [ICustomizable](#), [IEventEmitter](#).

Interface for the geometry editor.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IGeometryEditor()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>	The geometry being edited.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geometry editor.

**Events**

Name	Description
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">statechange</a>	Change to the geometry editor state. Instance of the <a href="#">Event</a> class.

**Methods**

Name	Description
<a href="#">startEditing()</a>	Enables editing mode.
<a href="#">stopEditing()</a>	Disables editing mode.

**Fields details****geometry**

```
{IGeometry} geometry
```

The geometry being edited.

**state**

```
{IDataManager} state
```

State of the geometry editor.

**Events details****statechange**

Change to the geometry editor state. Instance of the [Event](#) class.

**Methods details****startEditing**

```
{ } startEditing()
```

Enables editing mode.

**stopEditing**

```
{ } stopEditing()
```

Disables editing mode.

**IGeometryEditorChildModel**

Extends [IGeometryEditorModel](#).

Interface for the child data model.

[Constructor](#) | [Fields](#) | [Methods](#)

**Constructor**

```
IGeometryEditorChildModel(geometry, editor, pixels, parent)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IBaseGeometry</a>  The child geometry being edited. The <a href="#">IBaseGeometry</a> interface is not aware of the <a href="#">pixelgeometrychange</a> event, therefore, the pixel data is retrieved from the parent data model.
<a href="#">editor</a> *	—	Type: <a href="#">IGeometryEditor</a>  Link to the geometry editor.
<a href="#">pixels</a> *	—	Type: <a href="#">Number[]</a>  The model's pixel data.
<a href="#">parent</a> *	—	Type: <a href="#">IGeometryEditorModel</a>  The parent data model.

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">editor</a>	<a href="#">IGeometryEditor</a>	Geometry editor.
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IBaseGeometry</a>	Geometry of the model.

**Methods**

Name	Returns	Description
<a href="#">destroy()</a>		Destructor.  Inherited from <a href="#">IGeometryEditorModel</a> .

Name	Returns	Description
<a href="#">getParent()</a>	<a href="#">IGeometryEditorModel</a>	Returns the parent data model.
<a href="#">getPixels()</a>	Number[]	Returns the model's pixel data.  Inherited from <a href="#">IGeometryEditorModel</a> .
<a href="#">setPixels(pixels)</a>		Sets the model's pixel data.

### Fields details

#### editor

```
{IGeometryEditor} editor
```

Geometry editor.

#### geometry

```
{IBaseGeometry} geometry
```

Geometry of the model.

### Methods details

#### getParent

```
{IGeometryEditorModel} getParent()
```

**Returns** the parent data model.

#### setPixels

```
{ } setPixels(pixels)
```

Sets the model's pixel data.

#### Parameters:

Parameter	Default value	Description
<a href="#">pixels</a> *	—	Type: Number[]  Pixel data.

\* Mandatory parameter/option.

## IGeometryEditorModel

Extends [IEventEmitter](#).

Interface for the data model of the geometry editor.

[Constructor](#) | [Fields](#) | [Methods](#)

### Constructor

```
IGeometryEditorModel(geometry, editor)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IBaseGeometry</a> The geometry being edited.
<a href="#">editor</a> *	—	Type: <a href="#">IGeometryEditor</a> Link to the geometry editor.

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

**Methods**

Name	Returns	Description
<a href="#">destroy()</a>		Destructor.
<a href="#">getPixels()</a>	Number[]	Returns the model's pixel data.

**Methods details****destroy**

```
{ } destroy()
```

Destructor.

**getPixels**

```
{Number[]} getPixels()
```

**Returns** the model's pixel data.

**IGeometryEditorRootModel**

Extends [IGeometryEditorModel](#).

Interface for the root data model.

[Constructor](#) | [Fields](#) | [Methods](#)

**Constructor**

```
IGeometryEditorRootModel(geometry, editor)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IGeometry</a> The geometry being edited.
<a href="#">editor</a> *	—	Type: <a href="#">IGeometryEditor</a> Link to the geometry editor.

\* Mandatory parameter/option.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

### Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destructor. Inherited from <a href="#">IGeometryEditorModel</a> .
<a href="#">getPixels()</a>	Number[]	Returns the model's pixel data. Inherited from <a href="#">IGeometryEditorModel</a> .

## IGeometryJson

The interface of an object that describes a JSON representation of the geometry.

[Constructor](#) | [Fields](#)

### Constructor

```
IGeometryJson()
```

### Fields

Name	Type	Description
<a href="#">type</a>	String	ID of the geometry type.

### Fields details

#### type

```
{String} type
```

ID of the geometry type.

## IGeoObject

Extends [IChildOnMap](#), [ICustomizable](#), [IDomEventEmitter](#), [IParentOnMap](#).

Interface for a geo object.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IGeoObject ()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">IDataManager</a>	Geo object data.
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object.

### Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometrychange</a>	Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li> </ul>
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> Inherited from <a href="#">IParentOnMap</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul>

Name	Description
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	<p>Returns reference to the map.</p> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	<p>Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.</p>
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	<p>The method provides synchronous access to the overlay.</p>
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	<p>Returns link to the parent object, or null if the parent element was not set.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	<p>Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>

## Fields details

### geometry

```
{IGeometry|null} geometry
```

Geo object geometry.

### properties

```
{IDataManager} properties
```

Geo object data.

### state

```
{IDataManager} state
```

State of the geo object.

### Events details

#### geometrychange

Change to the geo object geometry. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- originalEvent: [IEvent](#) - Original event of the geometry.

#### overlaychange

Change to the geo object overlay. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- overlay: [IOverlay](#)|null - Reference to the overlay.
- oldOverlay: [IOverlay](#)|null - Previous overlay of the geo object.

#### propertieschange

Change to the geo object data. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- originalEvent: [IEvent](#) - Original event of the data manager.

### Methods details

#### getOverlay

```
{vow.Promise} getOverlay()
```

**Returns** the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.

#### getOverlaySync

```
{IOverlay|null} getOverlaySync()
```

The method provides synchronous access to the overlay.

**Returns** a reference to the overlay, or null if the overlay is missing.

## IGeoObjectCollection

Extends [ICustomizable](#), [IEventEmitter](#), [IParentOnMap](#).

Interface of a collection of geo objects.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IGeoObjectCollection()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

**Events**

Name	Description
<a href="#">add</a>	A child geo object has been added (inserted). Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>index: Integer - Index of the added geo object.</li> <li>child: <a href="#">IGeoObject</a> - Reference to the added geo object.</li> </ul>
<a href="#">boundschange</a>	Change to coordinates of the geographical area that spans the collection and its child geo objects. Instance of the <a href="#">Event</a> class.
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> Inherited from <a href="#">IParentOnMap</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">pixelboundschange</a>	Change to pixel coordinates of the area that includes the collection and its child geo objects. Instance of the <a href="#">Event</a> class.
<a href="#">remove</a>	A child geo object has been removed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>index: Integer - Index of the deleted geo object.</li> <li>child: <a href="#">IGeoObject</a> - Reference to the deleted geo object.</li> </ul>
<a href="#">set</a>	A new child geo object has been added to the collection. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>index: Integer - Index of the geo object.</li> <li>child: <a href="#">IGeoObject</a> - Reference to the new geo object.</li> <li>prevChild: <a href="#">IGeoObject</a> - Reference to the previous value for this index.</li> </ul>

**Methods**

Name	Returns	Description
<a href="#">add(child[, index])</a>	<a href="#">IGeoObjectCollection</a>	Adds (inserts) a child geo object to the collection.
<a href="#">each(callback[, context])</a>		Calls a handler function for each child geo object.
<a href="#">get(index)</a>	<a href="#">IGeoObject</a>	Returns a child geo object with the specified index.

Name	Returns	Description
<a href="#">getBounds()</a>	Number[][] null	Returns geographical coordinates of the area that covers the collection and its child geo objects.
<a href="#">getIterator()</a>	<a href="#">Iterator</a>	Returns iterator for the collection.
<a href="#">getLength()</a>	Integer	Returns length of the collection.
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getPixelBounds()</a>	Number[][] null	Returns global pixel coordinates of the area that spans the collection and its child geo objects.
<a href="#">indexOf(object)</a>	Integer	Returns index of the child geo object. If the geo object cannot be found in the collection, -1 is returned.
<a href="#">remove(child)</a>	<a href="#">IGeoObjectCollection</a>	Removes a child geo object from the collection.
<a href="#">removeAll()</a>	<a href="#">IGeoObjectCollection</a>	Clears the collection.
<a href="#">set(index, child)</a>	<a href="#">IGeoObjectCollection</a>	Adds a new child geo object to the collection.
<a href="#">splice(index, number)</a>	<a href="#">IGeoObjectCollection</a>	Removes geo objects from the collection. If necessary, puts other objects in their place. Objects that will be added in place of the deleted ones are passed as additional parameters (after the "number" parameter).

## Events details

### add

A child geo object has been added (inserted). Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- index: Integer - Index of the added geo object.
- child: [IGeoObject](#) - Reference to the added geo object.

### boundschange

Change to coordinates of the geographical area that spans the collection and its child geo objects. Instance of the [Event](#) class.

### pixelboundschange

Change to pixel coordinates of the area that includes the collection and its child geo objects. Instance of the [Event](#) class.

**remove**

A child geo object has been removed. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- index: Integer - Index of the deleted geo object.
- child: [IGeoObject](#) - Reference to the deleted geo object.

**set**

A new child geo object has been added to the collection. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- index: Integer - Index of the geo object.
- child: [IGeoObject](#) - Reference to the new geo object.
- prevChild: [IGeoObject](#) - Reference to the previous value for this index.

**Methods details****add**

```
{IGeoObjectCollection} add(child[, index])
```

Adds (inserts) a child geo object to the collection.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<a href="#">child</a> *	—	Type: <a href="#">IGeoObject</a> Child object.
<a href="#">index</a>	—	Type: Integer The index where the new object is added. By default, the object is added to the end of the collection.

\* Mandatory parameter/option.

**each**

```
{ } each(callback[, context])
```

Calls a handler function for each child geo object.

**Parameters:**

Parameter	Default value	Description
<a href="#">callback</a> *	—	Type: Function Handler function.
<a href="#">context</a>	—	Type: Object Context for the handler function.

\* Mandatory parameter/option.

## get

```
{IGeoObject} get(index)
```

**Returns** a child geo object with the specified index.

### Parameters:

Parameter	Default value	Description
<code>index</code> *	—	Type: Integer Index.

\* Mandatory parameter/option.

## getBounds

```
{Number[][]|null} getBounds()
```

**Returns** geographical coordinates of the area that covers the collection and its child geo objects.

## getIterator

```
{Iterator} getIterator()
```

**Returns** iterator for the collection.

## getLength

```
{Integer} getLength()
```

**Returns** length of the collection.

## getPixelBounds

```
{Number[][]|null} getPixelBounds()
```

**Returns** global pixel coordinates of the area that spans the collection and its child geo objects.

## indexOf

```
{Integer} indexOf(object)
```

**Returns** index of the child geo object. If the geo object cannot be found in the collection, -1 is returned.

### Parameters:

Parameter	Default value	Description
<code>object</code> *	—	Type: Object Child object.

\* Mandatory parameter/option.

## remove

```
{IGeoObjectCollection} remove(child)
```

Removes a child geo object from the collection.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>child *</code>	—	Type: <a href="#">IGeoObject</a> Geo object being removed.

\* Mandatory parameter/option.

### removeAll

```
{IGeoObjectCollection} removeAll()
```

Clears the collection.

**Returns** self-reference.

### set

```
{IGeoObjectCollection} set(index, child)
```

Adds a new child geo object to the collection.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>index *</code>	—	Type: Integer Index.
<code>child *</code>	—	Type: <a href="#">IGeoObject</a> Child object.

\* Mandatory parameter/option.

### splice

```
{IGeoObjectCollection} splice(index, number)
```

Removes geo objects from the collection. If necessary, puts other objects in their place. Objects that will be added in place of the deleted ones are passed as additional parameters (after the "number" parameter).

**Returns** collection of deleted geo objects.

**Parameters:**

Parameter	Default value	Description
<code>index *</code>	—	Type: Integer Index of the geo object to start deletion from.
<code>number *</code>	—	Type: Integer The number of geo objects to be deleted.

\* Mandatory parameter/option.

## IGeoObjectPopupData

Static object.

A data object that the geo object passes to its own info object (for example, to a balloon or hint).

### Fields

#### Fields

Name	Type	Description
<a href="#">geometry</a>	<a href="#">IGeometry</a>	Reference to the geometry of the geo object.
<a href="#">geoObject</a>	<a href="#">IGeoObject</a>	Reference to the geo object.
<a href="#">properties</a>	<a href="#">IDataManager</a>	Reference to the properties of the geo object.
<a href="#">userData</a>	Object	User data.

#### Fields details

##### geometry

```
{IGeometry} geometry
```

Reference to the geometry of the geo object.

##### geoObject

```
{IGeoObject} geoObject
```

Reference to the geo object.

##### properties

```
{IDataManager} properties
```

Reference to the properties of the geo object.

##### userData

```
{Object} userData
```

User data.

## IGeoObjectSequence

Extends [ICustomizable](#), [IEventEmitter](#), [IParentOnMap](#).

Interface of an unchangeable collection of geo objects.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IGeoObjectSequence()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

**Events**

Name	Description
<a href="#">boundschange</a>	Change to coordinates of the geographical area that spans the collection and its child geo objects. Instance of the <a href="#">Event</a> class.
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> Inherited from <a href="#">IParentOnMap</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">pixelboundschange</a>	Change to pixel coordinates of the area that includes the collection and its child geo objects. Instance of the <a href="#">Event</a> class.

**Methods**

Name	Returns	Description
<a href="#">each(callback[, context])</a>		Calls a handler function for each child geo object.
<a href="#">get(index)</a>	<a href="#">IGeoObject</a>	Returns a child geo object with the specified index.
<a href="#">getBounds()</a>	Number[][] null	Returns geographical coordinates of the area that covers the collection and its child geo objects.
<a href="#">getIterator()</a>	<a href="#">IIterator</a>	Returns iterator for child geo objects in the collection.
<a href="#">getLength()</a>	Integer	Returns length of the collection.
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getPixelBounds()</a>	Number[][] null	Returns global pixel coordinates of the area that spans the collection and its child geo objects.

Name	Returns	Description
<code>indexOf(object)</code>	Integer	Returns index of the child geo object. If the geo object cannot be found in the collection, -1 is returned.

### Events details

#### boundschange

Change to coordinates of the geographical area that spans the collection and its child geo objects. Instance of the [Event](#) class.

#### pixelboundschange

Change to pixel coordinates of the area that includes the collection and its child geo objects. Instance of the [Event](#) class.

### Methods details

#### each

```
{} each(callback[, context])
```

Calls a handler function for each child geo object.

#### Parameters:

Parameter	Default value	Description
<code>callback</code> *	—	Type: Function Handler function.
<code>context</code>	—	Type: Object Context for the handler function.

\* Mandatory parameter/option.

#### get

```
{IGeoObject} get(index)
```

Returns a child geo object with the specified index.

#### Parameters:

Parameter	Default value	Description
<code>index</code> *	—	Type: Integer Index.

\* Mandatory parameter/option.

#### getBounds

```
{Number[][]|null} getBounds()
```

**Returns** geographical coordinates of the area that covers the collection and its child geo objects.

### getIterator

```
{Iterator} getIterator()
```

**Returns** iterator for child geo objects in the collection.

### getLength

```
{Integer} getLength()
```

**Returns** length of the collection.

### getPixelBounds

```
{Number[][]|null} getPixelBounds()
```

**Returns** global pixel coordinates of the area that spans the collection and its child geo objects.

### indexOf

```
{Integer} indexOf(object)
```

**Returns** index of the child geo object. If the geo object cannot be found in the collection, -1 is returned.

#### Parameters:

Parameter	Default value	Description
<a href="#">object</a> *	—	Type: <a href="#">IGeoObject</a> Child object.

\* Mandatory parameter/option.

## IGroupControlLayout

Extends [ILayout](#).

Interface for the layout of a group control.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IGroupControlLayout(data)
```

#### Parameters:

Parameter	Default value	Description
<a href="#">data</a> *	—	Type: Object Layout data.

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .

**Events**

Name	Description
<a href="#">childcontainerchange</a>	Change to the container for child elements. Data fields: <ul style="list-style-type: none"> <li><code>newChildContainerElement</code> - New element for child objects.</li> <li><code>oldChildContainerElement</code> - Old element for child objects.</li> </ul>
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">emptinesschange</a>	Change to the empty layout indicator. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">ILayout</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseleave</a>	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mousemove</a>	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseup</a>	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">multitouchend</a>	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface.  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">multitouchmove</a>	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields: <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">multitouchstart</a>	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields: <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">parentelementchange</a>	Change to the parent element. Instance of the <a href="#">Event</a> class.  Inherited from <a href="#">ILayout</a> .
<a href="#">shapechange</a>	Change to the shape of the area spanning the layout. Instance of the <a href="#">Event</a> class.  Inherited from <a href="#">ILayout</a> .
<a href="#">wheel</a>	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destructor. Called when activity with the layout is finished.  Inherited from <a href="#">ILayout</a> .
<a href="#">getChildContainerElement()</a>	Object	Returns a reference to the DOM element that the child elements should be added to.
<a href="#">getData()</a>	Object	Returns layout data object.  Inherited from <a href="#">ILayout</a> .

Name	Returns	Description
<a href="#">getParentElement()</a>	HTMLElement	Returns parent HTML element. Inherited from <a href="#">ILayout</a> .
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element. Inherited from <a href="#">ILayout</a> .
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from <a href="#">ILayout</a> .
<a href="#">setData(data)</a>		Sets layout data. Inherited from <a href="#">ILayout</a> .
<a href="#">setParentElement(parent)</a>		Adds the layout to the DOM tree. Inherited from <a href="#">ILayout</a> .

## Events details

### childcontainerchange

Change to the container for child elements.

Data fields:

- `newChildContainerElement` - New element for child objects.
- `oldChildContainerElement` - Old element for child objects.

## Methods details

### getChildContainerElement

```
{Object} getChildContainerElement()
```

**Returns** a reference to the DOM element that the child elements should be added to.

## IHint

Extends [IPopup](#).

Interface for a hint.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IHint()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

**Events**

Name	Description
<a href="#">close</a>	Closing the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">open</a>	Opening the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .

**Methods**

Name	Returns	Description
<a href="#">close</a> ( <a href="#">[force]</a> )	<a href="#">vow.Promise</a>	Closes the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">getData</a> ()		Returns info object data. Inherited from <a href="#">IPopup</a> .
<a href="#">getOverlay</a> ()	<a href="#">vow.Promise</a>	Returns the promise object to return the overlay. Inherited from <a href="#">IPopup</a> .
<a href="#">getOverlaySync</a> ()	<a href="#">IOverlay</a>	Returns the overlay, if one exists. Inherited from <a href="#">IPopup</a> .
<a href="#">getPosition</a> ()		Returns the coordinates of the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">isOpen</a> ()	Boolean	Returns the info object state: open/closed. Inherited from <a href="#">IPopup</a> .

Name	Returns	Description
<code>open([position[, data]])</code>	<code>vow.Promise</code>	Opens the info object at the specified position. If the info object is already open, it moves it to the specified point. The format and content of the coordinates is determined by the <a href="#">IProjection</a> that is in the options.  Inherited from <a href="#">IPopup</a> .
<code>setData(data)</code>	<code>vow.Promise</code>	Defines new data for the info object.  Inherited from <a href="#">IPopup</a> .
<code>setPosition(position)</code>	<code>vow.Promise</code>	Specifies a new position for the info object.  Inherited from <a href="#">IPopup</a> .

## IHintManager

Extends [IPopupManager](#).

Interface for the hint manager.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IHintManager ()
```

### Fields

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

### Events

Name	Description
<code>close</code>	Closing the info object. Names of fields available via <a href="#">Event.get</a> : <ul style="list-style-type: none"> <li><code>target</code> - Reference to the object where the closing occurred.</li> </ul> Inherited from <a href="#">IPopupManager</a> .
<code>open</code>	Opening the info object. Names of fields available via <a href="#">Event.get</a> : <ul style="list-style-type: none"> <li><code>target</code> - Reference to the object where the opening occurred.</li> </ul> Inherited from <a href="#">IPopupManager</a> .

## Methods

Name	Returns	Description
<a href="#">close</a> ( <a href="#">[force]</a> )	<a href="#">vow.Promise</a>	Closes the info object. Inherited from <a href="#">IPopupManager</a> .
<a href="#">destroy</a> ()		Disables the info object manager. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getData</a> ()	Object null	Returns the data of the info object or 'null'. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOptions</a> ()	<a href="#">IOptionManager</a>  null	Returns the options manager or 'null'. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOverlay</a> ()	<a href="#">vow.Promise</a>	Returns the promise object to return the overlay. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOverlaySync</a> ()	<a href="#">IOverlay</a>  null	Returns the overlay, if one exists. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getPosition</a> ()	Number[] null	Returns the coordinates of the info object or 'null'. Inherited from <a href="#">IPopupManager</a> .
<a href="#">isOpen</a> ()	Boolean	Returns the info object state: open/closed. Inherited from <a href="#">IPopupManager</a> .
<a href="#">open</a> ( <a href="#">[[position</a> ], <a href="#">data</a> ], <a href="#">options</a> ]]])	<a href="#">vow.Promise</a>	Opens the info object at the specified position. Inherited from <a href="#">IPopupManager</a> .
<a href="#">setData</a> ( <a href="#">data</a> )	<a href="#">vow.Promise</a>	Defines new data for the info object. Inherited from <a href="#">IPopupManager</a> .
<a href="#">setOptions</a> ( <a href="#">options</a> )	<a href="#">vow.Promise</a>	Defines new options for the info object. Inherited from <a href="#">IPopupManager</a> .

Name	Returns	Description
<a href="#">setPosition(position)</a>	<a href="#">vow.Promise</a>	Specifies a new position for the info object.  Inherited from <a href="#">IPopupManager</a> .

## IHintOwner

An object with a hint can be accessed through the "hint" property.

[Constructor](#) | [Fields](#) | [Events](#)

### Constructor

```
IHintOwner()
```

### Fields

Name	Type	Description
<a href="#">hint</a>	<a href="#">IHintManager</a>	Object hint.

### Events

Name	Description
<a href="#">hintclose</a>	Hiding the hint.
<a href="#">hintopen</a>	Opening the hint.

### Fields details

#### hint

```
{IHintManager} hint
```

Object hint.

### Events details

#### hintclose

Hiding the hint.

#### hintopen

Opening the hint.

## IHotspot

Extends [IDomEventEmitter](#).

Interface of a shape that defines the hotspot geometry.

**Note:** The shape should be set in global pixel coordinates.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

IHotspot()

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .

**Events**

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseleave</a>	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mousemove</a>	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseup</a>	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">multitouchend</a>	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface. Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getShape()</a>	<a href="#">IShape</a>	Returns the hotspot shape.
<a href="#">getZIndex()</a>	Number	Returns <code>zIndex</code> of the hotspot.
<a href="#">setShape(shape)</a>		Sets the hotspot shape.
<a href="#">setZIndex(zIndex)</a>		Sets the z-index of the hotspot.

## Methods details

### getShape

```
{IShape} getShape()
```

Returns the hotspot shape.

### getZIndex

```
{Number} getZIndex()
```

Returns `zIndex` of the hotspot.

**setShape**

```
{ } setShape (shape)
```

Sets the hotspot shape.

**Parameters:**

Parameter	Default value	Description
<a href="#">shape</a> *	—	Type: <a href="#">IShape</a> Shape.

\* Mandatory parameter/option.

**setZIndex**

```
{ } setZIndex (zIndex)
```

Sets the z-index of the hotspot.

**Parameters:**

Parameter	Default value	Description
<a href="#">zIndex</a> *	—	Type: Number The z-index that will be set for the hotspot.

\* Mandatory parameter/option.

**IHotspotLayerObject**

Extends [ICustomizable](#), [IDomEventEmitter](#).

Interface for the hotspot layer object.

**Note:** A figure describing the hotspot should be set in global pixel coordinates.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
IHotspotLayerObject ()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape that defines a hotspot. Instance of the <a href="#">Event</a> class.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getGeometry()</a>	Object	Returns the actual geometry of an object.
<a href="#">getHotspot()</a>	<a href="#">IHotspot</a>	Returns object describing a hotspot.
<a href="#">getId()</a>	Number	Returns object ID.
<a href="#">getProperties()</a>	Object	Returns object data.
<a href="#">setGeometry(geometry)</a>		Defines the actual geometry of the object.
<a href="#">setId(id)</a>		Sets the object ID.
<a href="#">setProperties(properties)</a>		Sets the object's data.

## Events details

### shapechange

Change to the shape that defines a hotspot. Instance of the [Event](#) class.

## Methods details

### getGeometry

```
{Object} getGeometry()
```

**Returns** the actual geometry of an object.

### getHotspot

```
{IHotspot} getHotspot()
```

**Returns** object describing a hotspot.

### getId

```
{Number} getId()
```

**Returns** object ID.

### getProperties

```
{Object} getProperties()
```

**Returns** object data.

### setGeometry

```
{ } setGeometry(geometry)
```

Defines the actual geometry of the object.

#### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: Object  The actual geometry of an object.

\* Mandatory parameter/option.

### setId

```
{ } setId(id)
```

Sets the object ID.

#### Parameters:

Parameter	Default value	Description
<a href="#">id</a> *	—	Type: Number  Object ID.

\* Mandatory parameter/option.

### setProperties

```
{ } setProperties(properties)
```

Sets the object's data.

#### Parameters:

Parameter	Default value	Description
<a href="#">properties</a> *	—	Type: Object Object data.

\* Mandatory parameter/option.

## IHotspotObjectSource

Extends [ICustomizable](#).

Source of objects for the hotspot layers.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IHotspotObjectSource()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

### Events

Name	Description
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .

### Methods

Name	Description
<a href="#">cancelLastRequest()</a>	Cancels the last request for data.
<a href="#">requestObjects(layer, tileNumber, zoom, callback)</a>	Builds an array of <a href="#">IHotspotLayerObject</a> objects corresponding to a particular layer, tile, and map zoom level, and passes it to the callback function.

## Methods details

### cancelLastRequest

```
{ } cancelLastRequest ()
```

Cancels the last request for data.

### requestObjects

```
{ } requestObjects (layer, tileNumber, zoom, callback)
```

Builds an array of [IHotspotLayerObject](#) objects corresponding to a particular layer, tile, and map zoom level, and passes it to the callback function.

#### Parameters:

Parameter	Default value	Description
<a href="#">layer</a> *	—	Type: <a href="#">hotspot.Layer</a> Hotspot layer.
<a href="#">tileNumber</a> *	—	Type: <a href="#">Number[]</a> Tile coordinates.
<a href="#">zoom</a> *	—	Type: <a href="#">Number</a> Zoom level.
<a href="#">callback</a> *	—	Type: <a href="#">Function</a> Handler function.

\* Mandatory parameter/option.

## IHotspotShape

Extends [ICustomizable](#), [IDomEventEmitter](#).

Interface of a shape that defines the hotspot geometry.

**Note:** Not supported, beginning from version 2.1.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IHotspotShape ()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">containerchange</a>	<p>Change to the container. Instance of the <a href="#">Event</a> class.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape that defines a hotspot. Instance of the <a href="#">Event</a> class.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getContainer()</a>	<a href="#">IHotspotContainer</a>	Returns the hotspot container.
<a href="#">getGeometry()</a>	Object	Returns the actual shape geometry.
<a href="#">getId()</a>	Number	Returns object ID.
<a href="#">getProperties()</a>	Object	Returns object data.
<a href="#">getShape()</a>	<a href="#">IShape</a>	Returns shape describing a hotspot.
<a href="#">setGeometry(geometry)</a>		Sets the actual geometry of the shape.
<a href="#">setId(id)</a>		Sets the object ID.
<a href="#">setProperties(properties)</a>		Sets the object's data.

Name	Returns	Description
<a href="#">setShape(shape)</a>		

### Events details

#### containerchange

Change to the container. Instance of the [Event](#) class.

#### shapechange

Change to the shape that defines a hotspot. Instance of the [Event](#) class.

### Methods details

#### getContainer

```
{IHotspotContainer} getContainer()
```

**Returns** the hotspot container.

#### getGeometry

```
{Object} getGeometry()
```

**Returns** the actual shape geometry.

#### getId

```
{Number} getId()
```

**Returns** object ID.

#### getProperties

```
{Object} getProperties()
```

**Returns** object data.

#### getShape

```
{IShape} getShape()
```

**Returns** shape describing a hotspot.

#### setGeometry

```
{ } setGeometry(geometry)
```

Sets the actual geometry of the shape.

#### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: Object Actual shape geometry.

\* Mandatory parameter/option.

### setId

```
{ } setId(id)
```

Sets the object ID.

#### Parameters:

Parameter	Default value	Description
<code>id</code> *	—	Type: Number Object ID.

\* Mandatory parameter/option.

### setProperties

```
{ } setProperties(properties)
```

Sets the object's data.

#### Parameters:

Parameter	Default value	Description
<code>properties</code> *	—	Type: Object Object data.

\* Mandatory parameter/option.

### setShape

```
{ } setShape(shape)
```

#### Parameters:

Parameter	Default value	Description
<code>shape</code> *	—	Type: <a href="#">IShape</a> Shape describing a hotspot.

\* Mandatory parameter/option.

## IIterator

Interface for an iterator. The iterator allows you to iterate through all the items in a collection.

[Constructor](#) | [Fields](#) | [Methods](#)

#### Constructor

```
IIterator()
```

#### Fields

Name	Type	Description
<code>STOP_ITERATION</code>	Object	The object returned by the method <a href="#">IIterator.getNext</a> at the end of the iteration.

**Methods**

Name	Returns	Description
<a href="#">getNext()</a>	Object  <a href="#">IIterator.STOP_ITERATION</a>	Returns a reference to the next item in the collection or the <a href="#">IIterator.#STOP_ITERATION</a> object if the iteration is completed.

**Fields details****STOP\_ITERATION**

```
{Object} STOP_ITERATION
```

The object returned by the method [IIterator.getNext](#) at the end of the iteration.

**Methods details****getNext**

```
{Object|IIterator.STOP_ITERATION} getNext()
```

**Returns** a reference to the next item in the collection or the [IIterator.#STOP\\_ITERATION](#) object if the iteration is completed.

**ILayer**

Extends [IChildOnMap](#), [ICustomizable](#), [IEventEmitter](#).

Interface for a map layer.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
ILayer()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

**Events**

Name	Description
<a href="#">brightnesschange</a>	Layer brightness change event.
<a href="#">copyrightschange</a>	Event for changes to available copyright information.
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .

Name	Description
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .
<a href="#">tileloadchange</a>	Tile upload status change event. Data fields: <ul style="list-style-type: none"> <li>readyTileNumber - Number of ready tiles. A tile is considered ready when it is downloaded and rendered. Type: Number.</li> <li>totalTileNumber - Total number of visible tiles. Type: Number.</li> </ul>
<a href="#">zoomrangechange</a>	Event for changes to available information about the zoom level range.

## Methods

Name	Returns	Description
<a href="#">getBrightness()</a>	Number	Optional method.
<a href="#">getCopyrights(coords, zoom)</a>	<a href="#">vow.Promise</a>	Optional method. Requests information about copyrights at the specified point with the specified zoom.
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IChildOnMap</a> .
<a href="#">getZoomRange(point)</a>	<a href="#">vow.Promise</a>	Optional method. Checks the available range of zoom levels at the specified point. If there is data, the returned promise object will be resolved and will pass as a result an array of two numbers - the minimum and maximum zoom level available at the point. If there is no data, the promise is rejected with an error.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .

## Events details

### brightnesschange

Layer brightness change event.

### copyrightschange

Event for changes to available copyright information.

### tileloadchange

Tile upload status change event. Data fields:

- `readyTileNumber` - Number of ready tiles. A tile is considered ready when it is downloaded and rendered. Type: Number.
- `totalTileNumber` - Total number of visible tiles. Type: Number.

### zoomrangechange

Event for changes to available information about the zoom level range.

### Methods details

#### getBrightness

```
{Number} getBrightness()
```

Optional method.

**Returns** the layer brightness value from 0 to 1 (0 is zero brightness, and 1 is maximum brightness). The total brightness of the layers added to the map determines which color is selected for the logo and copyrights on the map.

#### getCopyrights

```
{vow.Promise} getCopyrights(coords, zoom)
```

Optional method. Requests information about copyrights at the specified point with the specified zoom.

**Returns** Promise that will be resolved and will pass as a result an array of strings or DOM elements with information about copyrights.

#### Parameters:

Parameter	Default value	Description
<code>coords</code> *	—	Type: Number[]  The point on the map that copyright information is being requested for.
<code>zoom</code> *	—	Type: Number  The zoom level that copyright information is being requested for.

\* Mandatory parameter/option.

#### getZoomRange

```
{vow.Promise} getZoomRange(point)
```

Optional method. Checks the available range of zoom levels at the specified point. If there is data, the returned promise object will be resolved and will pass as a result an array of two numbers - the minimum and maximum zoom level available at the point. If there is no data, the promise is rejected with an error.

**Returns** Promise object.

#### Parameters:

Parameter	Default value	Description
<a href="#">point</a> *	—	Type: Number[] Point

\* Mandatory parameter/option.

## ILayout

Extends [IDomEventEmitter](#).

Layout interface.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
ILayout (data)
```

#### Parameters:

Parameter	Default value	Description
<a href="#">data</a> *	—	Type: Object Layout data.

\* Mandatory parameter/option.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .

### Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">emptinesschange</a>	Change to the empty layout indicator. Instance of the <a href="#">Event</a> class.

Name	Description
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">parentelementchange</a>	<p>Change to the parent element. Instance of the <a href="#">Event</a> class.</p>

Name	Description
<a href="#">shapechange</a>	Change to the shape of the area spanning the layout. Instance of the <a href="#">Event</a> class.
<a href="#">wheel</a>	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destructor. Called when activity with the layout is finished.
<a href="#">getData()</a>	Object	Returns layout data object.
<a href="#">getParentElement()</a>	HTMLElement	Returns parent HTML element.
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element.
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.
<a href="#">setData(data)</a>		Sets layout data.
<a href="#">setParentElement(parent)</a>		Adds the layout to the DOM tree.

## Events details

### emptinesschange

Change to the empty layout indicator. Instance of the [Event](#) class.

### parentelementchange

Change to the parent element. Instance of the [Event](#) class.

### shapechange

Change to the shape of the area spanning the layout. Instance of the [Event](#) class.

## Methods details

### destroy

```
{ } destroy()
```

Destructor. Called when activity with the layout is finished.

### getData

```
{Object} getData()
```

Returns layout data object.

### getParentElement

```
{HTMLElement} getParentElement()
```

Returns parent HTML element.

### getShape

```
{IShape|null} getShape()
```

Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element.

### isEmpty

```
{Boolean} isEmpty()
```

Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.

### setData

```
{ } setData(data)
```

Sets layout data.

#### Parameters:

Parameter	Default value	Description
<code>data</code> *	—	Type: Object Layout data.

\* Mandatory parameter/option.

### setParentElement

```
{ } setParentElement(parent)
```

Adds the layout to the DOM tree.

#### Parameters:

Parameter	Default value	Description
<a href="#">parent</a> *	—	Type: HTMLElement null  Parent HTML element. The parent element must be added to the DOM tree. If null is passed, the element is removed from the DOM tree.

\* Mandatory parameter/option.

## ILinearRingGeometryAccess

Extends [IFreezable](#).

Interface for access to the "Closed contour" geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
ILinearRingGeometryAccess()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager for the object.  Inherited from <a href="#">IFreezable</a> .

### Events

Name	Description
<a href="#">change</a>	Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>oldCoordinates</code> - Old coordinates</li> <li><code>newCoordinates</code> - New coordinates.</li> <li><code>oldFillRule</code> - Old fill rule.</li> <li><code>newFillRule</code> - New fill rule.</li> </ul>

### Methods

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the contour.
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	Switches the object to "frozen" mode.  Inherited from <a href="#">IFreezable</a> .
<a href="#">get(index)</a>	Number[]	Returns coordinates of the point with the specified index.

Name	Returns	Description
<a href="#">getChildGeometry(index)</a>	<a href="#">IPointGeometryAccess</a>	Creates and returns an <a href="#">IPointGeometryAccess</a> object for the specified contour on the polyline.
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for a point on the contour closest to the anchorPosition.
<a href="#">getCoordinates()</a>	Number[][]	Returns an array of geometry coordinates.
<a href="#">getFillRule()</a>	String	Returns ID of the fill rule.
<a href="#">getLength()</a>	Integer	Returns the number of points in the geometry.
<a href="#">insert(index, coordinates)</a>	<a href="#">ILinearRingGeometryAccess</a>	Adds a new point with the specified index.
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false.  Inherited from <a href="#">IFreezable</a> .
<a href="#">remove(index)</a>	Number[]	Removes the point with the specified index.
<a href="#">set(index, coordinates)</a>	<a href="#">ILinearRingGeometryAccess</a>	Sets coordinates of the point with the specified index.
<a href="#">setCoordinates(coordinates)</a>	<a href="#">ILinearRingGeometryAccess</a>	Sets an array of geometry coordinates.
<a href="#">setFillRule(fillRule)</a>	<a href="#">ILinearRingGeometryAccess</a>	Sets the contour fill rule.
<a href="#">splice(index, number)</a>	Number[][]	Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter.
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode.  Inherited from <a href="#">IFreezable</a> .

### Events details

#### change

Changed coordinates. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- oldCoordinates - Old coordinates
- newCoordinates - New coordinates.
- oldFillRule - Old fill rule.
- newFillRule - New fill rule.

## Methods details

### contains

```
{Boolean} contains(position)
```

Checks whether the passed point is located inside the contour.

**Returns** an indicator for whether the point belongs to the contour.

#### Parameters:

Parameter	Default value	Description
<a href="#">position</a> *	—	Type: Number[] Coordinates of a point.

\* Mandatory parameter/option.

### get

```
{Number[]} get(index)
```

**Returns** coordinates of the point with the specified index.

#### Parameters:

Parameter	Default value	Description
<a href="#">index</a> *	—	Type: Integer Index of a point.

\* Mandatory parameter/option.

### getChildGeometry

```
{IPointGeometryAccess} getChildGeometry(index)
```

Creates and returns an [IPointGeometryAccess](#) object for the specified contour on the polyline.

**Returns** the "Point" geometry object that corresponds to the specified endpoint.

#### Parameters:

Parameter	Default value	Description
<a href="#">index</a> *	—	Type: Integer Index of a contour vertex.

\* Mandatory parameter/option.

### getClosest

```
{Object} getClosest(anchorPosition)
```

Searches for a point on the contour closest to the `anchorPosition`.

**Returns** an object with the following fields:

- `position` - Point on the contour closest to "anchorPosition".
- `distance` - Distance from "anchorPosition" to "position".

- `closestPointIndex` - Index of the vertex closest to "position".
- `nextPointIndex` - Index of the vertex that follows "position".
- `prevPointIndex` - Index of the vertex that precedes "position".

The "nextPointIndex" and "prevPointIndex" fields may be omitted if "position" coincides with one of the contour vertexes.

#### Parameters:

Parameter	Default value	Description
<code>anchorPosition</code> *	—	Type: Number[]  Coordinates of a point for which the nearest contour vertex is calculated.

\* Mandatory parameter/option.

#### getCoordinates

```
{Number[][]} getCoordinates()
```

Returns an array of geometry coordinates.

#### getFillRule

```
{String} getFillRule()
```

Returns ID of the fill rule.

#### getLength

```
{Integer} getLength()
```

Returns the number of points in the geometry.

#### insert

```
{ILinearRingGeometryAccess} insert(index, coordinates)
```

Adds a new point with the specified index.

Returns self-reference.

#### Parameters:

Parameter	Default value	Description
<code>index</code> *	—	Type: Integer  Index of a point.
<code>coordinates</code> *	—	Type: Number[]  Coordinates of a point.

\* Mandatory parameter/option.

#### remove

```
{Number[]} remove(index)
```

Removes the point with the specified index.

**Returns** the coordinates of a deleted point.

**Parameters:**

Parameter	Default value	Description
<code>index *</code>	—	Type: Integer Index of a point.

\* Mandatory parameter/option.

**set**

```
{ILinearRingGeometryAccess} set(index, coordinates)
```

Sets coordinates of the point with the specified index.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>index *</code>	—	Type: Integer Index of a point.
<code>coordinates *</code>	—	Type: Number[] Coordinates of a point.

\* Mandatory parameter/option.

**setCoordinates**

```
{ILinearRingGeometryAccess} setCoordinates(coordinates)
```

Sets an array of geometry coordinates.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>coordinates *</code>	—	Type: Number[][] Geometry coordinates.

\* Mandatory parameter/option.

**setFillRule**

```
{ILinearRingGeometryAccess} setFillRule(fillRule)
```

Sets the contour fill rule.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>fillRule</code> *	—	Type: String ID of the fill rule.

\* Mandatory parameter/option.

## splice

```
{Number[][]} splice(index, number)
```

Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter.

**Returns** an array of coordinates of deleted points.

### Parameters:

Parameter	Default value	Description
<code>index</code> *	—	Type: Integer The index to start from for removing and adding points.
<code>number</code> *	—	Type: Integer The number of deleted points.

\* Mandatory parameter/option.

## ILineStringGeometry

Extends [IGeometry](#), [ILineStringGeometryAccess](#).

Interface for the "Polyline" geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
ILineStringGeometry()
```

### Fields

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<code>options</code>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">change</a>	<p>Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>oldCoordinates - Old coordinates</li> <li>newCoordinates - New coordinates.</li> </ul> <p>Inherited from <a href="#">ILineStringGeometryAccess</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> <p>Inherited from <a href="#">IGeometry</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">pixelgeometrychange</a>	<p>The pixel geometry changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>pixelGeometry - New <a href="#">IPixelGeometry</a> pixel geometry.</li> </ul> <p>Inherited from <a href="#">IGeometry</a>.</p>

## Methods

Name	Returns	Description
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	<p>Switches the object to "frozen" mode.</p> <p>Inherited from <a href="#">IFreezable</a>.</p>
<a href="#">get(index)</a>	Number[]	<p>Returns coordinates of the point with the specified index.</p> <p>Inherited from <a href="#">ILineStringGeometryAccess</a>.</p>
<a href="#">getBounds()</a>	Number[][] null	<p>Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner.</p> <p>Inherited from <a href="#">IGeometry</a>.</p>
<a href="#">getChildGeometry(index)</a>	<a href="#">IPointGeometryAccess</a>	<p>Creates and returns an <a href="#">IPointGeometryAccess</a> object for the specified vertex on the polyline.</p> <p>Inherited from <a href="#">ILineStringGeometryAccess</a>.</p>
<a href="#">getClosest(anchorPosition)</a>	Object	<p>Searches for a point on the polyline closest to the anchorPosition.</p> <p>Inherited from <a href="#">ILineStringGeometryAccess</a>.</p>

Name	Returns	Description
<a href="#">getCoordinates()</a>	Number[][]	Returns an array of geometry coordinates.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">getLength()</a>	Integer	Returns the number of points in the geometry.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">getMap()</a>	Map null	Returns the current map.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getPixelGeometry([options])</a>	<a href="#">IPixelGeometry</a>	Returns the pixel geometry corresponding to the given geometry, its options, and the map state.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getType()</a>	String	Returns the "LineString" string.
<a href="#">insert(index, coordinates)</a>	<a href="#">ILineStringGeometryAccess</a>	Adds a new point with the specified index.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false.  Inherited from <a href="#">IFreezable</a> .
<a href="#">remove(index)</a>	Number[]	Removes the point with the specified index.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">set(index, coordinates)</a>	<a href="#">ILineStringGeometryAccess</a>	Sets coordinates of the point with the specified index.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">setCoordinates(coordinates)</a>	<a href="#">ILineStringGeometryAccess</a>	Sets an array of geometry coordinates.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">setMap(map)</a>		Sets the map.  Inherited from <a href="#">IGeometry</a> .

Name	Returns	Description
<a href="#">splice(index, number)</a>	Number[][]	Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter.  Inherited from <a href="#">ILineStringGeometryAccess</a> .
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode.  Inherited from <a href="#">IFreezable</a> .

### Methods details

#### getType

```
{String} getType()
```

Returns the "LineString" string.

## ILineStringGeometryAccess

Extends [IFreezable](#).

Interface for access to the "Polyline" geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
ILineStringGeometryAccess()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager for the object.  Inherited from <a href="#">IFreezable</a> .

### Events

Name	Description
<a href="#">change</a>	Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>oldCoordinates</code> - Old coordinates</li> <li><code>newCoordinates</code> - New coordinates.</li> </ul>

**Methods**

Name	Returns	Description
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	Switches the object to "frozen" mode.  Inherited from <a href="#">IFreezable</a> .
<a href="#">get(index)</a>	Number[]	Returns coordinates of the point with the specified index.
<a href="#">getChildGeometry(index)</a>	<a href="#">IPointGeometryAccess</a>	Creates and returns an <a href="#">IPointGeometryAccess</a> object for the specified vertex on the polyline.
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for a point on the polyline closest to the anchorPosition.
<a href="#">getCoordinates()</a>	Number[][]	Returns an array of geometry coordinates.
<a href="#">getLength()</a>	Integer	Returns the number of points in the geometry.
<a href="#">insert(index, coordinates)</a>	<a href="#">ILineStringGeometryAccess</a>	Adds a new point with the specified index.
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false.  Inherited from <a href="#">IFreezable</a> .
<a href="#">remove(index)</a>	Number[]	Removes the point with the specified index.
<a href="#">set(index, coordinates)</a>	<a href="#">ILineStringGeometryAccess</a>	Sets coordinates of the point with the specified index.
<a href="#">setCoordinates(coordinates)</a>	<a href="#">ILineStringGeometryAccess</a>	Sets an array of geometry coordinates.
<a href="#">splice(index, number)</a>	Number[][]	Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter.
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode.  Inherited from <a href="#">IFreezable</a> .

**Events details****change**

Changed coordinates. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- oldCoordinates - Old coordinates

- `newCoordinates` - New coordinates.

## Methods details

### get

```
{Number[]} get(index)
```

**Returns** coordinates of the point with the specified index.

#### Parameters:

Parameter	Default value	Description
<code>index</code> *	—	Type: Integer Index of a point.

\* Mandatory parameter/option.

#### Example:

```
// Marking the start of the line with a placemark:
map.geoObjects.add(
  new ymaps.Placemark(polyline.geometry.get(0), { iconContent: 'A' })
);
```

### getChildGeometry

```
{IPointGeometryAccess} getChildGeometry(index)
```

Creates and returns an [IPointGeometryAccess](#) object for the specified vertex on the polyline.

**Returns** the "Point" geometry object that corresponds to the specified endpoint.

#### Parameters:

Parameter	Default value	Description
<code>index</code> *	—	Type: Integer Index of an endpoint.

\* Mandatory parameter/option.

### getClosest

```
{Object} getClosest(anchorPosition)
```

Searches for a point on the polyline closest to the `anchorPosition`.

**Returns** an object with the following fields:

- `position` - Point on the polyline closest to "anchorPosition".
- `distance` - Distance from "anchorPosition" to "position".
- `closestPointIndex` - Index of the vertex closest to "position".
- `nextPointIndex` - Index of the vertex that follows "position".
- `prevPointIndex` - Index of the vertex that precedes "position".

The "nextPointIndex" and "prevPointIndex" fields may be omitted if "position" coincides with one of the polyline vertexes.

#### Parameters:

Parameter	Default value	Description
<code>anchorPosition</code> *	—	Type: Number[]  Coordinates of a point for which the nearest polyline vertex is calculated.

\* Mandatory parameter/option.

#### Example:

```
// Deleting points from the line when clicked:
myPolyline.events.add('click', function (e) {
  myPolyline.geometry.remove(
    myPolyline.geometry.getClosest(e.get('coords')).closestPointIndex
  );
});
```

#### getCoordinates

```
{Number[][]} getCoordinates()
```

Returns an array of geometry coordinates.

#### getLength

```
{Integer} getLength()
```

Returns the number of points in the geometry.

#### insert

```
{ILineStringGeometryAccess} insert(index, coordinates)
```

Adds a new point with the specified index.

Returns self-reference.

#### Parameters:

Parameter	Default value	Description
<code>index</code> *	—	Type: Integer  Index of a point.
<code>coordinates</code> *	—	Type: Number[]  Coordinates of a point.

\* Mandatory parameter/option.

#### Example:

```
// Adding a new point to the end of the line when the map is clicked.
myMap.events.add('click', function (e) {
  myLineString.insert(myLineString.getLength(), e.get('coords'))
});
```

#### remove

```
{Number[]} remove(index)
```

Removes the point with the specified index.

**Returns** the coordinates of a deleted point.

**Parameters:**

Parameter	Default value	Description
<code>index *</code>	—	Type: Integer Index of a point.

\* Mandatory parameter/option.

**set**

```
{ILineStringGeometryAccess} set(index, coordinates)
```

Sets coordinates of the point with the specified index.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>index *</code>	—	Type: Integer Index of a point.
<code>coordinates *</code>	—	Type: Number[] Coordinates of a point.

\* Mandatory parameter/option.

**setCoordinates**

```
{ILineStringGeometryAccess} setCoordinates(coordinates)
```

Sets an array of geometry coordinates.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>coordinates *</code>	—	Type: Number[][] Geometry coordinates.

\* Mandatory parameter/option.

**splice**

```
{Number[][]} splice(index, number)
```

Deletes a defined number of points, starting from the specified index. New points can be added in place of the deleted ones. Coordinates of the new points can be passed as additional arguments after the "number" parameter.

**Returns** an array of coordinates of deleted points.

**Parameters:**

Parameter	Default value	Description
<code>index</code> *	—	Type: Integer  The index to start from for removing and adding points.
<code>number</code> *	—	Type: Integer  The number of deleted points.

\* Mandatory parameter/option.

#### Example:

```
// Adding a new point to the start of the line when the map is clicked.
myMap.events.add('click', function (e) {
  myLineString.splice(0, 0, myLineString.getLength(), e.get('coords'))
});
```

## IMapAction

Extends [IEventEmitter](#).

Interface of an object that manages map movement.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IMapAction()
```

### Fields

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

### Events

Name	Description
<code>end</code>	Event that notifies the map that movement has finished.
<code>tick</code>	Event that notifies the map of the next step. Contains the fields: <ul style="list-style-type: none"> <li><code>globalPixelCenter</code> - The new map center, in global pixels.</li> <li><code>zoom</code> - The new map zoom.</li> <li><code>duration</code> - The time that is allowed for performing the step.</li> <li><code>timingFunction</code> - Function describing the type of movement.</li> </ul>

### Methods

Name	Description
<code>begin(mapActionManager)</code>	Starts the movement to be performed by the map. This method is called automatically by the map movement manager. From the moment when <a href="#">IMapAction.begin</a> is called, the movement manager listens for <a href="#">IMapAction.event:tick</a> and <a href="#">IMapAction.event:end</a> and executes them.

Name	Description
<a href="#">end()</a>	Stops movement.

### Events details

#### end

Event that notifies the map that movement has finished.

#### tick

Event that notifies the map of the next step. Contains the fields:

- `globalPixelCenter` - The new map center, in global pixels.
- `zoom` - The new map zoom.
- `duration` - The time that is allowed for performing the step.
- `timingFunction` - Function describing the type of movement.

### Methods details

#### begin

```
{ } begin(mapActionManager)
```

Starts the movement to be performed by the map. This method is called automatically by the map movement manager. From the moment when [IMapAction.begin](#) is called, the movement manager listens for [IMapAction.event:tick](#) and [IMapAction.event:end](#) and executes them.

#### Parameters:

Parameter	Default value	Description
<a href="#">mapActionManager</a> *	—	Type: <a href="#">map.action.Manager</a>  The movement manager for the map that movement is being performed on.

\* Mandatory parameter/option.

#### end

```
{ } end()
```

Stops movement.

## IMapObjectCollection

Extends [ICollection](#), [ICustomizable](#), [IParentOnMap](#).

Collection of objects on the map.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IMapObjectCollection()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

**Events**

Name	Description
<a href="#">add</a>	A child object was added. Inherited from <a href="#">ICollection</a> .
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> Inherited from <a href="#">IParentOnMap</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">remove</a>	A child object was deleted. Inherited from <a href="#">ICollection</a> .

**Methods**

Name	Returns	Description
<a href="#">add(object)</a>	<a href="#">ICollection</a>	Adds a child object to the collection. Inherited from <a href="#">ICollection</a> .
<a href="#">getIterator()</a>	<a href="#">IIterator</a>	Returns iterator for the collection. Inherited from <a href="#">ICollection</a> .
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">remove(object)</a>	<a href="#">ICollection</a>	Removes a child object from the collection. Inherited from <a href="#">ICollection</a> .

**IMapState**

Object descriptor of the map state. Objects with this interface are returned by the YMapsML loader service.

See [geoXml.load](#)

[Constructor](#) | [Methods](#)

**Constructor**

```
IMapState()
```

**Methods**

Name	Returns	Description
<a href="#">applyToMap(map)</a>	<a href="#">vow.Promise</a>	Applies the state to the map that was passed.

**Methods details****applyToMap**

```
{vow.Promise} applyToMap(map)
```

Applies the state to the map that was passed.

**Returns** Promise object.

**Parameters:**

Parameter	Default value	Description
<a href="#">map</a> *	—	Type: <a href="#">Map</a> Map.

\* Mandatory parameter/option.

**IMultiRouteModelJson**

Interface of the model description object of a multi-stop route.

[Constructor](#) | [Fields](#)

**Constructor**

```
IMultiRouteModelJson()
```

**Fields**

Name	Type	Description
<a href="#">params</a>	<a href="#">IMultiRouteParams</a>	Routing options.
<a href="#">referencePoints</a>	<a href="#">IMultiRouteReferencePoint[]</a>	The description of the reference points on the multi-stop route. Also see the description of the parameter <a href="#">IMultiRouteParams.vialIndexes</a> .

**Fields details****params**

```
{IMultiRouteParams} params
```

Routing options.

**referencePoints**

```
{IMultiRouteReferencePoint[]} referencePoints
```

The description of the reference points on the multi-stop route. Also see the description of the parameter [IMultiRouteParams.viaIndexes](#).

## IMultiRouteParams

Interface of the object that describes the format of the parameters of a multi-stop route model.

[Constructor](#) | [Fields](#)

### Constructor

```
IMultiRouteParams()
```

### Fields

Name	Type	Description
<a href="#">avoidTrafficJams</a>	Boolean	Allows to construct a multi-stop route with consideration for the current traffic. Default value: false.
<a href="#">boundedBy</a>	Number[][] null	Defines the area on the map where the objects being searched for are presumably located. This is used if the route points are set using the mailing address, and not coordinates. Default value: null.
<a href="#">requestSendInterval</a>	String Number	Time interval between requests to the router service. Can be set in milliseconds, otherwise the optimal value will be calculated automatically. Default value: "auto".
<a href="#">results</a>	Integer	The maximum number of routes returned by the multi-stop router. Default value is 3.
<a href="#">reverseGeocoding</a>	Boolean	Whether to use reverse geocoding for points specified as coordinates.
<a href="#">routingMode</a>	String	Routing type. Accepts one of the two string values: <ul style="list-style-type: none"> <li>"auto" - Driving route.</li> <li>"masstransit" - Routing using public transport.</li> <li>"pedestrian" - Walking route.</li> <li>"bicycle" - Bicycle route.</li> </ul> Default value: "auto".
<a href="#">searchCoordOrder</a>	String	Determines how to interpret descriptions of the reference points that can be defined either as arrays of coordinates or as geometries. Can take one of two values: "longlat" or "latlong". The current value of the API's <a href="#">coordorder parameter</a> is used by default.
<a href="#">strictBounds</a>	Boolean	To search for objects only within the area specified by the boundedBy parameter. This is used if the route points are set using the mailing address, and not coordinates. Default value: false.
<a href="#">viaIndexes</a>	Integer[]	Contains indexes of throughpoints in the set of reference points of the multi-stop route. The array is empty by default.

### Fields details

#### avoidTrafficJams

```
{Boolean} avoidTrafficJams
```

Allows to construct a multi-stop route with consideration for the current traffic. Default value: false.

### boundedBy

```
{Number[][]|null} boundedBy
```

Defines the area on the map where the objects being searched for are presumably located. This is used if the route points are set using the mailing address, and not coordinates. Default value: null.

### requestSendInterval

```
{String|Number} requestSendInterval
```

Time interval between requests to the router service. Can be set in milliseconds, otherwise the optimal value will be calculated automatically. Default value: "auto".

### results

```
{Integer} results
```

The maximum number of routes returned by the multi-stop router. Default value is 3.

### reverseGeocoding

```
{Boolean} reverseGeocoding
```

Whether to use reverse geocoding for points specified as coordinates.

### routingMode

```
{String} routingMode
```

Routing type. Accepts one of the two string values:

- "auto" - Driving route.
- "masstransit" - Routing using public transport.
- "pedestrian" - Walking route.
- "bicycle" - Bicycle route.

Default value: "auto".

### searchCoordOrder

```
{String} searchCoordOrder
```

Determines how to interpret descriptions of the reference points that can be defined either as arrays of coordinates or as geometries. Can take one of two values: "longlat" or "latlong". The current value of the API's [coordorder parameter](#) is used by default.

### strictBounds

```
{Boolean} strictBounds
```

To search for objects only within the area specified by the boundedBy parameter. This is used if the route points are set using the mailing address, and not coordinates. Default value: false.

### viaIndexes

```
{Integer[]} viaIndexes
```

Contains indexes of throughpoints in the set of reference points of the multi-stop route. The array is empty by default.

## IMultiRouteReferencePoint

Interface of the object that describes the format of a multi-stop route reference point.

### Constructor

### Constructor

```
IMultiRouteReferencePoint()
```

Each reference point of a multi-stop route can be set using one of the following ways:

- A string containing the postal address of the reference point.
- An array containing the latitude and longitude of the reference point.
- A `geometry.Point` geometry describing the reference point.

Reference points on a multi-stop route can be either waypoints or throughpoints. A waypoint signifies a stop, and waypoints divide the route into so-called paths. A throughpoint is not a stop. Thus, when passing via a throughpoint, the segment of the route path won't break.

## IMultiRouterRouteBalloon

**Note:** The constructor of the `IMultiRouterRouteBalloon` class is hidden, as this class is not intended for autonomous initialization.

Interface of the model description object of a multi-stop route.

[Events](#) | [Methods](#)

### Events

Name	Description
<a href="#">close</a>	Closing the balloon.
<a href="#">open</a>	Opening the balloon.

### Methods

Name	Returns	Description
<a href="#">close()</a>	<a href="#">vow.Promise</a>	Closes route's balloon.
<a href="#">isOpen()</a>	Boolean	Returns the info object state: open/closed.
<a href="#">open([position])</a>	<a href="#">vow.Promise</a>	Opens route's balloon at closest position on route to specified location. Also makes route active.

### Events details

#### close

Closing the balloon.

#### open

Opening the balloon.

## Methods details

### close

```
{vow.Promise} close()
```

Closes route's balloon.

**Returns** Promise object.

### isOpen

```
{Boolean} isOpen()
```

**Returns** the info object state: open/closed.

### open

```
{vow.Promise} open([position])
```

Opens route's balloon at closest position on route to specified location. Also makes route active.

**Returns** Promise object.

#### Parameters:

Parameter	Default value	Description
<a href="#">position</a>	—	Type: Number[]  The point where you want to place the balloon. By default balloon will open in the middle of the route.

## IOptionManager

Extends [IChild](#), [IEventEmitter](#), [IFreezable](#).

Interface for the options manager. The options manager lets you set option values, build an options inheritance hierarchy, and resolve option values in the context of the existing inheritance hierarchy.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IOptionManager()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager for the object.  Inherited from <a href="#">IFreezable</a> .

### Events

Name	Description
<a href="#">change</a>	Changes to the option.

Name	Description
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>

## Methods

Name	Returns	Description
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	<p>Switches the object to "frozen" mode.</p> <p>Inherited from <a href="#">IFreezable</a>.</p>
<a href="#">get(key[, defaultValue])</a>		<p>Returns the value of the specified option in the context of the existing options inheritance hierarchy. When this method is called, first values are searched for in the current options manager, then, if the value is not defined, the search continues in the hierarchy of parent managers.</p>
<a href="#">getAll()</a>	Object	<p>Returns a reference to the internal hash that stores option values.</p>
<a href="#">getName()</a>	String	<p>Returns name of the options manager.</p>
<a href="#">getNative(key)</a>	Object	<p>Returns the value of the specified option that is defined for the given level of the options hierarchy, i.e. in this manager.</p>
<a href="#">getParent()</a>	<a href="#">IOptionsManager</a>  null	<p>Returns parent options manager.</p>
<a href="#">isFrozen()</a>	Boolean	<p>Returns true if the object is in "frozen" mode, otherwise false.</p> <p>Inherited from <a href="#">IFreezable</a>.</p>
<a href="#">resolve(key[, name])</a>	Object	<p>Method intended to be called by child options managers.</p>
<a href="#">setName(name)</a>		<p>Sets the name of the options manager.</p>
<a href="#">setParent(parent)</a>	<a href="#">IChild</a>	<p>Sets the parent options manager.</p>

Name	Returns	Description
<code>unfreeze()</code>	<a href="#">IFreezable</a>	Switches the object to active mode. Inherited from <a href="#">IFreezable</a> .

### Events details

#### change

Changes to the option.

### Methods details

#### get

```
{ } get(key[, defaultValue])
```

**Returns** the value of the specified option in the context of the existing options inheritance hierarchy. When this method is called, first values are searched for in the current options manager, then, if the value is not defined, the search continues in the hierarchy of parent managers.

#### Parameters:

Parameter	Default value	Description
<code>key</code> *	—	Type: String Name of the option.
<code>defaultValue</code>	—	Type: Object Default value.

\* Mandatory parameter/option.

#### getAll

```
{Object} getAll()
```

**Returns** a reference to the internal hash that stores option values.

#### getName

```
{String} getName()
```

**Returns** name of the options manager.

#### getNative

```
{Object} getNative(key)
```

**Returns** the value of the specified option that is defined for the given level of the options hierarchy, i.e. in this manager.

#### Parameters:

Parameter	Default value	Description
<code>key *</code>	—	Type: String Name of the option.

\* Mandatory parameter/option.

### getParent

```
{IOptionManager|null} getParent()
```

Returns parent options manager.

### resolve

```
{Object} resolve(key[, name])
```

Method intended to be called by child options managers.

Returns the option's value in the parent context.

#### Parameters:

Parameter	Default value	Description
<code>key *</code>	—	Type: String Name of the option.
<code>name</code>	—	Type: String Name of the child options manager.

\* Mandatory parameter/option.

### setName

```
{ } setName(name)
```

Sets the name of the options manager.

#### Parameters:

Parameter	Default value	Description
<code>name *</code>	—	Type: String Name of the options manager.

\* Mandatory parameter/option.

### setParent

```
{IChild} setParent(parent)
```

Sets the parent options manager.

Returns self-reference.

#### Parameters:

Parameter	Default value	Description
<a href="#">parent</a> *	—	Type: <a href="#">IOptionManager</a>  null Parent options manager.

\* Mandatory parameter/option.

## IOverlay

Extends [ICustomizable](#), [IDomEventEmitter](#).

Interface for an overlay.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IOverlay()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

### Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">datachange</a>	Data change. Data fields: <ul style="list-style-type: none"> <li><code>oldData</code> - Old data.</li> <li><code>newData</code> - New data.</li> </ul>
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">emptinesschange</a>	Change to the empty overlay flag. Instance of the <a href="#">Event</a> class.
<a href="#">geometrychange</a>	Changed geometry. Data fields: <ul style="list-style-type: none"> <li><code>oldGeometry</code> - Old pixel geometry.</li> <li><code>newGeometry</code> - New pixel geometry.</li> </ul>

Name	Description
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"><li>• <code>oldMap</code> - Old map.</li><li>• <code>newMap</code> - New map.</li></ul>
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseleave</a>	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mousemove</a>	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseup</a>	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">multitouchend</a>	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">multitouchmove</a>	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields: <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the overlay. Instance of the <a href="#">Event</a> class.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getData()</a>	Object	Returns the overlay data object.
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	Returns the current pixel geometry.
<a href="#">getMap()</a>	<a href="#">Map</a>  null	Returns reference to the current map.
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape.
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.
<a href="#">setData(data)</a>		Sets the overlay data.
<a href="#">setGeometry(geometry)</a>		Sets the overlay pixel geometry.
<a href="#">setMap(map)</a>		Sets the map on which to display the overlay.

## Events details

### datachange

Data change. Data fields:

- `oldData` - Old data.
- `newData` - New data.

### emptinesschange

Change to the empty overlay flag. Instance of the [Event](#) class.

### geometrychange

Changed geometry. Data fields:

- `oldGeometry` - Old pixel geometry.
- `newGeometry` - New pixel geometry.

### mapchange

Map reference changed. Data fields:

- `oldMap` - Old map.
- `newMap` - New map.

### shapechange

Change to the shape of the area spanning the overlay. Instance of the [Event](#) class.

## Methods details

### getData

```
{Object} getData()
```

**Returns** the overlay data object.

### getGeometry

```
{IPixelGeometry} getGeometry()
```

**Returns** the current pixel geometry.

### getMap

```
{Map|null} getMap()
```

**Returns** reference to the current map.

### getShape

```
{IShape|null} getShape()
```

**Returns** a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape.

### isEmpty

```
{Boolean} isEmpty()
```

**Returns** true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.

### setData

```
{ } setData (data)
```

Sets the overlay data.

#### Parameters:

Parameter	Default value	Description
<a href="#">data</a> *	—	Type: Object Overlay data.

\* Mandatory parameter/option.

### setGeometry

```
{ } setGeometry (geometry)
```

Sets the overlay pixel geometry.

#### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IPixelGeometry</a> The geometry in global pixel coordinates.

\* Mandatory parameter/option.

### setMap

```
{ } setMap (map)
```

Sets the map on which to display the overlay.

#### Parameters:

Parameter	Default value	Description
<a href="#">map</a> *	—	Type: <a href="#">Map</a>  null Reference to the map.

\* Mandatory parameter/option.

## IPane

Extends [IEventEmitter](#).

Interface for a map pane. The map pane is an object that hosts its DOM element with a certain zIndex inside the map container. A pane can serve as a container to hold representations of various map elements, such as tiles, overlays, static map controls, etc. A pane can also act as a non-transparent screen for DOM events where you can listen to mouse events.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

IPane()

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

**Events**

Name	Description
<a href="#">overflowchange</a>	Change to the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. Instance of <a href="#">IEvent</a> .
<a href="#">zindexchange</a>	Change to the zIndex value of a pane. Instance of <a href="#">IEvent</a> .

**Methods**

Name	Returns	Description
<a href="#">destroy()</a>		Destroys the pane.
<a href="#">getElement()</a>	HTMLElement	Returns a reference to the pane's DOM container.
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns the map that the pane belongs to.
<a href="#">getOverflow()</a>	String	Returns value of the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. This parameter can take one of the following string values: <ul style="list-style-type: none"> <li>"visible" - When you go off the map container, the content of the pane remains visible.</li> <li>"hidden" - The viewport for the pane content is limited by the map container.</li> </ul>
<a href="#">getZIndex()</a>	Number	Returns the zIndex of the pane.

**Events details****overflowchange**

Change to the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. Instance of [IEvent](#).

**zindexchange**

Change to the zIndex value of a pane. Instance of [IEvent](#).

## Methods details

### destroy

```
{ } destroy()
```

Destroys the pane.

### getElement

```
{HTMLElement} getElement()
```

Returns a reference to the pane's DOM container.

### getMap

```
{Map} getMap()
```

Returns the map that the pane belongs to.

### getOverflow

```
{String} getOverflow()
```

Returns value of the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. This parameter can take one of the following string values:

- "visible" - When you go off the map container, the content of the pane remains visible.
- "hidden" - The viewport for the pane content is limited by the map container.

### getZIndex

```
{Number} getZIndex()
```

Returns the zIndex of the pane.

## IPanorama

Interface for describing a panorama.

[Constructor](#) | [Methods](#)

### Constructor

```
IPanorama()
```

### Methods

Name	Returns	Description
<a href="#">getAngularBBox()</a>	Number[]	Returns spherical coordinates that define the area covered by the image on the panoramic sphere. Coordinates are specified in the format <code>[thetaTop, phiRight, thetaBottom, phiLeft]</code> (the same as CSS).

Name	Returns	Description
<a href="#">getConnectionArrows()</a>	<a href="#">IPanoramaConnectionArrow</a> []	Returns array of connection arrows on the panorama.
<a href="#">getConnectionMarkers()</a>	<a href="#">IPanoramaConnectionMarker</a> []	Returns array of connection markers on the panorama.
<a href="#">getCoordSystem()</a>	<a href="#">ICoordSystem</a>	Returns the coordinate system that is used for defining positions of the panorama and all the associated markers and connections.
<a href="#">getDefaultDirection()</a>	Number[]	Returns default direction of view. It will be used by the player when opening the panorama.
<a href="#">getDefaultSpan()</a>	Number[]	Returns default size of the viewing area. It will be used by the player when opening the panorama.
<a href="#">getGraph()</a>	<a href="#">IPanoramaGraph</a>  null	Returns the graph of panoramas connected to the current panorama for making quick transitions.
<a href="#">getMarkers()</a>	<a href="#">IPanoramaMarker</a> []	Returns array of markers on the panorama.
<a href="#">getName()</a>	String	Returns the name of the panorama displayed by the player in the interface.
<a href="#">getPosition()</a>	Number[]	Returns the location of the panorama in the coordinate system set in the options. Set in the format [lon, lat, height], [lat, lon, height] or [x, y, height] depending on the coordinate system and order. height – the height of the panorama in meters, relative to some level (not necessarily sea level).
<a href="#">getTileLevels()</a>	<a href="#">IPanoramaTileLevel</a> []	Returns array of zoom levels for the panorama image.
<a href="#">getTileSize()</a>	Number[]	Returns the size of tiles that the panorama image is divided into.

## Methods details

### getAngularBBox

```
{Number[]} getAngularBBox()
```

**Returns** spherical coordinates that define the area covered by the image on the panoramic sphere. Coordinates are specified in the format `[thetaTop, phiRight, thetaBottom, phiLeft]` (the same as CSS).

### getConnectionArrows

```
{IPanoramaConnectionArrow[]} getConnectionArrows()
```

**Returns** array of connection arrows on the panorama.

### getConnectionMarkers

```
{IPanoramaConnectionMarker[]} getConnectionMarkers()
```

**Returns** array of connection markers on the panorama.

### getCoordSystem

```
{ICoordSystem} getCoordSystem()
```

**Returns** the coordinate system that is used for defining positions of the panorama and all the associated markers and connections.

### getDefaultDirection

```
{Number[]} getDefaultDirection()
```

**Returns** default direction of view. It will be used by the player when opening the panorama.

### getDefaultSpan

```
{Number[]} getDefaultSpan()
```

**Returns** default size of the viewing area. It will be used by the player when opening the panorama.

### getGraph

```
{IPanoramaGraph|null} getGraph()
```

**Returns** the graph of panoramas connected to the current panorama for making quick transitions.

### getMarkers

```
{IPanoramaMarker[]} getMarkers()
```

**Returns** array of markers on the panorama.

### getName

```
{String} getName()
```

**Returns** the name of the panorama displayed by the player in the interface.

### getPosition

```
{Number[]} getPosition()
```

**Returns** the location of the panorama in the coordinate system set in the options. Set in the format `[lon, lat, height]`, `[lat, lon, height]` or `[x, y, height]` depending on the coordinate system and order. `height` – the height of the panorama in meters, relative to some level (not necessarily sea level).

### getTileLevels

```
{IPanoramaTileLevel[]} getTileLevels()
```

Returns array of zoom levels for the panorama image.

### getTileSize

```
{Number[]} getTileSize()
```

Returns the size of tiles that the panorama image is divided into.

## IPanoramaConnection

Interface describing an entity on the panorama that connects it to another panorama.

[Constructor](#) | [Methods](#)

### Constructor

```
IPanoramaConnection()
```

### Methods

Name	Returns	Description
<a href="#">getConnectedPanorama()</a>	<a href="#">vow.Promise</a>	Returns the promise object that will be resolved by the connected panorama object or rejected with an error.

### Methods details

#### getConnectedPanorama

```
{vow.Promise} getConnectedPanorama()
```

Returns the promise object that will be resolved by the connected panorama object or rejected with an error.

## IPanoramaConnectionArrow

Extends [IPanoramaConnection](#).

Interface describing a connection to another panorama that's rendered as a clickable arrow.

[Constructor](#) | [Fields](#) | [Methods](#)

### Constructor

```
IPanoramaConnectionArrow()
```

### Fields

Name	Type	Description
<a href="#">properties</a>	<a href="#">data.Manager</a>	Additional properties of the panorama connection.

**Methods**

Name	Returns	Description
<a href="#">getConnectedPanorama()</a>	<a href="#">vow.Promise</a>	Returns the promise object that will be resolved by the connected panorama object or rejected with an error.  Inherited from <a href="#">IPanoramaConnection</a> .
<a href="#">getDirection()</a>	Number[]	Returns the direction to the panorama indicated by the connection from the panorama it belongs to.
<a href="#">getPanorama()</a>	<a href="#">IPanorama</a>	Returns the panorama the connection belongs to.

**Fields details****properties**

```
{data.Manager} properties
```

Additional properties of the panorama connection.

**Methods details****getDirection**

```
{Number[]} getDirection()
```

**Returns** the direction to the panorama indicated by the connection from the panorama it belongs to.

**getPanorama**

```
{IPanorama} getPanorama()
```

**Returns** the panorama the connection belongs to.

**IPanoramaConnectionMarker**

Extends [IPanoramaConnection](#), [IPanoramaMarker](#).

Interface that describes panorama connections.

[Constructor](#) | [Fields](#) | [Methods](#)

**Constructor**

```
IPanoramaConnectionMarker()
```

**Fields**

Name	Type	Description
<a href="#">properties</a>	<a href="#">data.Manager</a>	Additional marker properties.  Inherited from <a href="#">IPanoramaMarker</a> .

**Methods**

Name	Returns	Description
<a href="#">getConnectedPanorama()</a>	<a href="#">vow.Promise</a>	Returns the promise object that will be resolved by the connected panorama object or rejected with an error.  Inherited from <a href="#">IPanoramaConnection</a> .
<a href="#">getIconSet()</a>	<a href="#">vow.Promise</a>	Returns a promise object that will be resolved by an object that implements the interface <a href="#">IPanoramaMarkerIconSet</a> and contains images for all the marker states.  Inherited from <a href="#">IPanoramaMarker</a> .
<a href="#">getPanorama()</a>	<a href="#">IPanorama</a>	Returns the panorama the marker belongs to.  Inherited from <a href="#">IPanoramaMarker</a> .
<a href="#">getPosition()</a>	<a href="#">Number[]</a>	Returns the marker's position in the coordinate system of the panorama that the graph containing the node belongs to. Set in the format <code>[lon, lat, height]</code> , <code>[lat, lon, height]</code> or <code>[x, y, height]</code> depending on the coordinate system and order. <code>height</code> – the height of the marker in meters, set relative to the same level as the height of the panorama.  Inherited from <a href="#">IPanoramaMarker</a> .

**IPanoramaGraph**

Interface that describes panorama graphs.

[Constructor](#) | [Methods](#)

**Constructor**

```
IPanoramaGraph()
```

**Methods**

Name	Returns	Description
<a href="#">getEdges()</a>	<a href="#">IPanoramaGraphEdge[]</a>	Returns an array of graph edges.
<a href="#">getNodes()</a>	<a href="#">IPanoramaGraphNode[]</a>	Returns an array of graph nodes.

Name	Returns	Description
<a href="#">getPanorama()</a>	<a href="#">IPanorama</a>	Returns the panorama the graph belongs to.

### Methods details

#### getEdges

```
{IPanoramaGraphEdge[]} getEdges()
```

Returns an array of graph edges.

#### getNodes

```
{IPanoramaGraphNode[]} getNodes()
```

Returns an array of graph nodes.

#### getPanorama

```
{IPanorama} getPanorama()
```

Returns the panorama the graph belongs to.

## IPanoramaGraphEdge

Interface describing an edge of the panorama graph.

[Constructor](#) | [Methods](#)

### Constructor

```
IPanoramaGraphEdge()
```

### Methods

Name	Returns	Description
<a href="#">getEndNodes()</a>	<a href="#">IPanoramaGraphNode[]</a>	Returns an array of two nodes of the graph connected by an edge.

### Methods details

#### getEndNodes

```
{IPanoramaGraphNode[]} getEndNodes()
```

Returns an array of two nodes of the graph connected by an edge.

## IPanoramaGraphNode

Extends [IPanoramaConnection](#).

Interface describing a node of the panorama graph.

[Constructor](#) | [Methods](#)

**Constructor**

```
IPanoramaGraphNode()
```

**Methods**

Name	Returns	Description
<a href="#">getConnectionPanorama()</a>	<a href="#">vow.Promise</a>	Returns the promise object that will be resolved by the connected panorama object or rejected with an error.  Inherited from <a href="#">IPanoramaConnection</a> .

**IPanoramaMarker**

Interface describing markers on the panorama.

[Constructor](#) | [Fields](#) | [Methods](#)

**Constructor**

```
IPanoramaMarker()
```

**Fields**

Name	Type	Description
<a href="#">properties</a>	<a href="#">data.Manager</a>	Additional marker properties.

**Methods**

Name	Returns	Description
<a href="#">getIconSet()</a>	<a href="#">vow.Promise</a>	Returns a promise object that will be resolved by an object that implements the interface <a href="#">IPanoramaMarkerIconSet</a> and contains images for all the marker states.
<a href="#">getPanorama()</a>	<a href="#">IPanorama</a>	Returns the panorama the marker belongs to.
<a href="#">getPosition()</a>	<a href="#">Number[]</a>	Returns the marker's position in the coordinate system of the panorama that the graph containing the node belongs to. Set in the format [lon, lat, height], [lat, lon, height] or [x, y, height] depending on the coordinate system and order. height – the height of the marker in meters, set relative to the same level as the height of the panorama.

## Fields details

### properties

```
{data.Manager} properties
```

Additional marker properties.

## Methods details

### getIconSet

```
{vow.Promise} getIconSet()
```

**Returns** a promise object that will be resolved by an object that implements the interface [IPanoramaMarkerIconSet](#) and contains images for all the marker states.

### getPanorama

```
{IPanorama} getPanorama()
```

**Returns** the panorama the marker belongs to.

### getPosition

```
{Number[]} getPosition()
```

**Returns** the marker's position in the coordinate system of the panorama that the graph containing the node belongs to. Set in the format `[lon, lat, height]`, `[lat, lon, height]` or `[x, y, height]` depending on the coordinate system and order. `height` – the height of the marker in meters, set relative to the same level as the height of the panorama.

## IPanoramaMarkerIcon

Interface describing the marker's icon.

[Constructor](#) | [Fields](#)

### Constructor

```
IPanoramaMarkerIcon()
```

### Fields

Name	Type	Description
<a href="#">image</a>	HTMLCanvasElement, HTMLImageElement	Icon image.
<a href="#">offset</a>	Number[]	Offset of the top left corner of the icon in pixels, relative to the projection of the point that the marker is bound to.

## Fields details

### image

```
{HTMLCanvasElement|HTMLImageElement} image
```

Icon image.

**offset**

```
{Number[]} offset
```

Offset of the top left corner of the icon in pixels, relative to the projection of the point that the marker is bound to.

**IPanoramaMarkerIconSet**

Interface describing a set of marker icons.

[Constructor](#) | [Fields](#)

**Constructor**

```
IPanoramaMarkerIconSet()
```

**Fields**

Name	Type	Description
<a href="#">default</a>	<a href="#">IPanoramaMarkerIcon</a>	Default state icon.
<a href="#">expanded</a>	<a href="#">IPanoramaMarkerIcon</a> null	Icon for the "active" marker state. The marker switches to this state when clicked; when clicked again, the marker becomes inactive.
<a href="#">expandedHovered</a>	<a href="#">IPanoramaMarkerIcon</a> null	Icon for the "active highlighted" state. The marker switches to this state when it is active and the cursor is on it.
<a href="#">hovered</a>	<a href="#">IPanoramaMarkerIcon</a> null	Icon for the "highlighted" marker state. The marker switches to this state when the cursor is on it.

**Fields details****default**

```
{IPanoramaMarkerIcon} default
```

Default state icon.

**expanded**

```
{IPanoramaMarkerIcon|null} expanded
```

Icon for the "active" marker state. The marker switches to this state when clicked; when clicked again, the marker becomes inactive.

**expandedHovered**

```
{IPanoramaMarkerIcon|null} expandedHovered
```

Icon for the "active highlighted" state. The marker switches to this state when it is active and the cursor is on it.

**hovered**

```
{IPanoramaMarkerIcon|null} hovered
```

Icon for the "highlighted" marker state. The marker switches to this state when the cursor is on it.

**IPanoramaTileLevel**

Interface for describing zoom levels of the panorama image.

[Constructor](#) | [Methods](#)

## Constructor

```
IPanoramaTileLevel()
```

## Methods

Name	Returns	Description
<a href="#">getImageSize()</a>	Number[]	Returns the size of the entire image in pixels.
<a href="#">getTileUrl(x, y)</a>	String	Gets the URL of the tile from its position in the panorama image. The position is set by a pair of indexes: horizontal and vertical. The horizontal index starts from the left at zero and increases to the right. The vertical index starts from the top of the image and increases going down.

## Methods details

### getImageSize

```
{Number[]} getImageSize()
```

**Returns** the size of the entire image in pixels.

### getTileUrl

```
{String} getTileUrl(x, y)
```

Gets the URL of the tile from its position in the panorama image. The position is set by a pair of indexes: horizontal and vertical. The horizontal index starts from the left at zero and increases to the right. The vertical index starts from the top of the image and increases going down.

**Returns** the URL of the tile with the passed indexes.

#### Parameters:

Parameter	Default value	Description
<a href="#">x</a> *	—	Type: Number Horizontal index of the tile.
<a href="#">y</a> *	—	Type: Number Vertical index of the tile.

\* Mandatory parameter/option.

## IParentOnMap

Interface for a parent object that belongs to a particular map object.

[Constructor](#) | [Events](#) | [Methods](#)

**Constructor**

```
IParentOnMap()
```

**Events**

Name	Description
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul>

**Methods**

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map.

**Events details****mapchange**

Map reference changed. Data fields:

- oldMap - Old map.
- newMap - New map.

**Methods details****getMap**

```
{Map} getMap()
```

**Returns** reference to the map.

**IPixelCircleGeometry**

Extends [IPixelGeometry](#).

Interface for the "Circle" pixel geometry.

[Constructor](#) | [Fields](#) | [Methods](#)

**Constructor**

```
IPixelCircleGeometry()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

**Methods**

Name	Returns	Description
<a href="#">equals(geometry)</a>	Boolean	Returns true if the passed geometry is equivalent to the given one.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getCoordinates()</a>	Number[]	Returns coordinates of the center of the circle.
<a href="#">getMetaData()</a>	Object	Returns metadata of the pixel geometry.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getRadius()</a>	Number	Returns radius of the circle.
<a href="#">getType()</a>	String	Returns ID of the geometry type.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">scale(factor)</a>	<a href="#">IPixelGeometry</a>	Creates a scaled copy of the geometry.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">shift(offset)</a>	<a href="#">IPixelGeometry</a>	Creates a copy of the geometry that is shifted by the specified amount.  Inherited from <a href="#">IPixelGeometry</a> .

**Methods details****getCoordinates**

```
{Number[]} getCoordinates()
```

**Returns** coordinates of the center of the circle.

**getRadius**

```
{Number} getRadius()
```

**Returns** radius of the circle.

## IPixelGeometry

Extends [IBaseGeometry](#).

Interface of a pixel geometry.

[Constructor](#) | [Fields](#) | [Methods](#)

### Constructor

```
IPixelGeometry()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

### Methods

Name	Returns	Description
<a href="#">equals(geometry)</a>	Boolean	Returns true if the passed geometry is equivalent to the given one.
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getMetaData()</a>	Object	Returns metadata of the pixel geometry.
<a href="#">getType()</a>	String	Returns ID of the geometry type.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">scale(factor)</a>	<a href="#">IPixelGeometry</a>	Creates a scaled copy of the geometry.
<a href="#">shift(offset)</a>	<a href="#">IPixelGeometry</a>	Creates a copy of the geometry that is shifted by the specified amount.

## Methods details

### equals

```
{Boolean} equals(geometry)
```

**Returns** true if the passed geometry is equivalent to the given one.

#### Parameters:

Parameter	Default value	Description
<code>geometry</code> *	—	Type: <a href="#">IPixelGeometry</a> The geometry to compare.

\* Mandatory parameter/option.

### getMetaData

```
{Object} getMetaData()
```

**Returns** metadata of the pixel geometry.

### scale

```
{IPixelGeometry} scale(factor)
```

Creates a scaled copy of the geometry.

**Returns** a scaled copy of the geometry.

#### Parameters:

Parameter	Default value	Description
<code>factor</code> *	—	Type: Number Scaling factor.

\* Mandatory parameter/option.

#### Example:

```
// Reducing the geometry to half its size  
var smallCopy = myPixelGeometry.scale(0.5);
```

### shift

```
{IPixelGeometry} shift(offset)
```

Creates a copy of the geometry that is shifted by the specified amount.

**Returns** a shifted copy of the geometry.

#### Parameters:

Parameter	Default value	Description
<code>offset</code> *	—	Type: Number[] Amount to shift on the axes.

\* Mandatory parameter/option.

**Example:**

```
// Shifting all the geometry's coordinates 200 pixels to the left
var shifted = myPixelGeometry.shift([-200, 0]);
```

**IPixelLineStringGeometry**

Extends [IPixelGeometry](#).

Interface for the "Polyline" pixel geometry.

[Constructor](#) | [Fields](#) | [Methods](#)

**Constructor**

```
IPixelLineStringGeometry()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

**Methods**

Name	Returns	Description
<a href="#">equals(geometry)</a>	Boolean	Returns true if the passed geometry is equivalent to the given one.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for a point on the polyline closest to the anchorPosition.
<a href="#">getCoordinates()</a>	Number[][]	Returns coordinates of a line.
<a href="#">getLength()</a>	Integer	Returns the number of points in the geometry.
<a href="#">getMetaData()</a>	Object	Returns metadata of the pixel geometry.  Inherited from <a href="#">IPixelGeometry</a> .

Name	Returns	Description
<code>getType()</code>	String	Returns ID of the geometry type. Inherited from <a href="#">IBaseGeometry</a> .
<code>scale(factor)</code>	<a href="#">IPixelGeometry</a>	Creates a scaled copy of the geometry. Inherited from <a href="#">IPixelGeometry</a> .
<code>shift(offset)</code>	<a href="#">IPixelGeometry</a>	Creates a copy of the geometry that is shifted by the specified amount. Inherited from <a href="#">IPixelGeometry</a> .

## Methods details

### getClosest

```
{Object} getClosest(anchorPosition)
```

Searches for a point on the polyline closest to the anchorPosition.

**Returns** an object with the following fields:

- position - Point on the polyline closest to "anchorPosition".
- distance - Distance from "anchorPosition" to "position".
- closestPointIndex - Index of the vertex closest to "position".
- nextPointIndex - Index of the vertex that follows "position".
- prevPointIndex - Index of the vertex that precedes "position".

The "nextPointIndex" and "prevPointIndex" fields may be omitted if "position" coincides with one of the polyline vertexes.

#### Parameters:

Parameter	Default value	Description
<code>anchorPosition</code> *	—	Type: Number[]  Coordinates of a point for which the nearest polyline vertex is calculated.

\* Mandatory parameter/option.

### getCoordinates

```
{Number[][]} getCoordinates()
```

**Returns** coordinates of a line.

### getLength

```
{Integer} getLength()
```

**Returns** the number of points in the geometry.

## IPixelMultiLineGeometry

Extends [IPixelGeometry](#).

Interface for the "Multiline" pixel geometry.

[Constructor](#) | [Fields](#) | [Methods](#)

### Constructor

```
IPixelMultiLineGeometry()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

### Methods

Name	Returns	Description
<a href="#">equals(geometry)</a>	Boolean	Returns true if the passed geometry is equivalent to the given one. Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for a point on the contour closest to the anchorPosition.
<a href="#">getCoordinates()</a>	Number[][][]	Returns coordinates of a multiline.
<a href="#">getLength()</a>	Integer	Returns the number of lines in the multiline.
<a href="#">getMetaData()</a>	Object	Returns metadata of the pixel geometry. Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getType()</a>	String	Returns ID of the geometry type. Inherited from <a href="#">IBaseGeometry</a> .

Name	Returns	Description
<code>scale(factor)</code>	<a href="#">IPixelGeometry</a>	Creates a scaled copy of the geometry.  Inherited from <a href="#">IPixelGeometry</a> .
<code>shift(offset)</code>	<a href="#">IPixelGeometry</a>	Creates a copy of the geometry that is shifted by the specified amount.  Inherited from <a href="#">IPixelGeometry</a> .

## Methods details

### getClosest

```
{Object} getClosest(anchorPosition)
```

Searches for a point on the contour closest to the anchorPosition.

**Returns** an object with the following fields:

- position - The point on the multipolygon contour that is nearest to "anchorPosition".
- distance - Distance from "anchorPosition" to "position".
- closestPointIndex - Index of the multipolygon vertex closest to "position".
- nextPointIndex - Index of the multipolygon vertex that follows "position".
- prevPointIndex - Index of the multipolygon vertex that precedes "position".
- pathIndex - Index of the multipolygon contour that the found point is associated with.

The "nextPointIndex" and "prevPointIndex" fields may be omitted if "position" coincides with one of the multipolygon vertexes.

#### Parameters:

Parameter	Default value	Description
<code>anchorPosition</code> *	—	Type: <code>Number[]</code>  Coordinates of a point for which the nearest contour vertex is calculated.

\* Mandatory parameter/option.

### getCoordinates

```
{Number[][][]} getCoordinates()
```

**Returns** coordinates of a multiline.

### getLength

```
{Integer} getLength()
```

**Returns** the number of lines in the multiline.

## IPixelMultiPolygonGeometry

Extends [IPixelGeometry](#).

Interface for the "Multipolygon" pixel geometry.

[Constructor](#) | [Fields](#) | [Methods](#)

## Constructor

```
IPixelMultiPolygonGeometry()
```

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

## Methods

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the multipolygon.
<a href="#">equals(geometry)</a>	Boolean	Returns true if the passed geometry is equivalent to the given one.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for the point nearest to "anchorPosition" on the multipolygon contour.
<a href="#">getCoordinates()</a>	Number[][][]	Returns coordinates of the multipolygon.

Name	Returns	Description
<a href="#">getFillRule()</a>	String	Returns the string ID that defines the multipolygon fill rule. The ID can have one of two values: <ul style="list-style-type: none"><li>• <code>evenOdd</code> - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it.</li><li>• <code>nonZero</code> - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.</li></ul>
<a href="#">getLength()</a>	Integer	Returns the number of polygons in the multipolygon.
<a href="#">getMetaData()</a>	Object	Returns metadata of the pixel geometry.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getType()</a>	String	Returns ID of the geometry type.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">scale(factor)</a>	<a href="#">IPixelGeometry</a>	Creates a scaled copy of the geometry.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">shift(offset)</a>	<a href="#">IPixelGeometry</a>	Creates a copy of the geometry that is shifted by the specified amount.  Inherited from <a href="#">IPixelGeometry</a> .

## Methods details

### contains

```
{Boolean} contains(position)
```

Checks whether the passed point is located inside the multipolygon.

**Returns** indicator for whether the point belongs to the multipolygon.

#### Parameters:

Parameter	Default value	Description
<code>position</code> *	—	Type: Number[] Coordinates of a point.

\* Mandatory parameter/option.

### getClosest

```
{Object} getClosest(anchorPosition)
```

Searches for the point nearest to "anchorPosition" on the multipolygon contour.

**Returns** an object with the following fields:

- `position` - The point on the multipolygon contour that is nearest to "anchorPosition".
- `distance` - Distance from "anchorPosition" to "position".
- `closestPointIndex` - Index of the multipolygon vertex closest to "position".
- `nextPointIndex` - Index of the multipolygon vertex that follows "position".
- `prevPointIndex` - Index of the multipolygon vertex that precedes "position".
- `pathIndex` - Index of the multipolygon contour that the found point is associated with.

The "nextPointIndex" and "prevPointIndex" fields may be omitted if "position" coincides with one of the multipolygon vertexes.

#### Parameters:

Parameter	Default value	Description
<code>anchorPosition</code> *	—	Type: Number[] Coordinates of a point for which the nearest vertex on the polygon contour is calculated.

\* Mandatory parameter/option.

### getCoordinates

```
{Number[][][][]} getCoordinates()
```

**Returns** coordinates of the multipolygon.

### getFillRule

```
{String} getFillRule()
```

**Returns** the string ID that defines the multipolygon fill rule. The ID can have one of two values:

- `evenOdd` - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it.
- `nonZero` - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.

### getLength

```
{Integer} getLength()
```

**Returns** the number of polygons in the multipolygon.

## IPixelPointGeometry

Extends [IPixelGeometry](#).

Interface for the "Point" pixel geometry.

[Constructor](#) | [Fields](#) | [Methods](#)

### Constructor

```
IPixelPointGeometry()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

### Methods

Name	Returns	Description
<a href="#">equals(geometry)</a>	Boolean	Returns true if the passed geometry is equivalent to the given one.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getCoordinates()</a>	Number[]	Returns coordinates of a point.

Name	Returns	Description
<a href="#">getMetaData()</a>	Object	Returns metadata of the pixel geometry. Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getType()</a>	String	Returns ID of the geometry type. Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">scale(factor)</a>	<a href="#">IPixelGeometry</a>	Creates a scaled copy of the geometry. Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">shift(offset)</a>	<a href="#">IPixelGeometry</a>	Creates a copy of the geometry that is shifted by the specified amount. Inherited from <a href="#">IPixelGeometry</a> .

## Methods details

### getCoordinates

```
{Number[]} getCoordinates()
```

Returns coordinates of a point.

## IPixelPolygonGeometry

Extends [IPixelGeometry](#).

Interface for the "Polygon" pixel geometry.

[Constructor](#) | [Fields](#) | [Methods](#)

### Constructor

```
IPixelPolygonGeometry()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

### Methods

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the polygon.

Name	Returns	Description
<a href="#">equals(geometry)</a>	Boolean	Returns true if the passed geometry is equivalent to the given one.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for the point nearest to "anchorPosition" on the polygon contour.
<a href="#">getCoordinates()</a>	Number[][][]	Returns coordinates of the polygon.
<a href="#">getFillRule()</a>	String	Returns string ID that defines the polygon fill rule. The ID accepts one of two values: <ul style="list-style-type: none"><li>• evenOdd - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it.</li><li>• nonZero - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.</li></ul>

Name	Returns	Description
<a href="#">getLength()</a>	Integer	Returns the number of contours in the polygon.
<a href="#">getMetaData()</a>	Object	Returns metadata of the pixel geometry. Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getType()</a>	String	Returns ID of the geometry type. Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">scale(factor)</a>	<a href="#">IPixelGeometry</a>	Creates a scaled copy of the geometry. Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">shift(offset)</a>	<a href="#">IPixelGeometry</a>	Creates a copy of the geometry that is shifted by the specified amount. Inherited from <a href="#">IPixelGeometry</a> .

## Methods details

### contains

```
{Boolean} contains(position)
```

Checks whether the passed point is located inside the polygon.

**Returns** indicator for whether the point belongs to the polygon.

#### Parameters:

Parameter	Default value	Description
<a href="#">position</a> *	—	Type: Number[] Coordinates of a point.

\* Mandatory parameter/option.

### getClosest

```
{Object} getClosest(anchorPosition)
```

Searches for the point nearest to "anchorPosition" on the polygon contour.

**Returns** an object with the following fields:

- `position` - The point on the polygon contour that is nearest to "anchorPosition".
- `distance` - Distance from "anchorPosition" to "position".
- `closestPointIndex` - Index of the polygon vertex closest to "position".
- `nextPointIndex` - Index of the polygon vertex that follows "position".
- `prevPointIndex` - Index of the polygon vertex that precedes "position".
- `pathIndex` - Index of the polygon contour that the found point is associated with.

The "nextPointIndex" and "prevPointIndex" fields may be omitted if "position" coincides with one of the polygon vertexes.

#### Parameters:

Parameter	Default value	Description
<a href="#">anchorPosition</a> *	—	Type: Number[]  Coordinates of a point for which the nearest vertex on the polygon contour is calculated.

\* Mandatory parameter/option.

#### getCoordinates

```
{Number[][][]} getCoordinates()
```

**Returns** coordinates of the polygon.

#### getFillRule

```
{String} getFillRule()
```

**Returns** string ID that defines the polygon fill rule. The ID accepts one of two values:

- `evenOdd` - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and counting the number of contour segments in this shape that are crossed by this ray. If the number is odd, the point is inside it; if even, the point is outside it.
- `nonZero` - Algorithm that checks whether a point is located in the fill area by drawing a ray from this point to infinity in any direction, and checking the points at which a segment of the shape crosses this ray. Starting from zero, one is added each time a segment crosses the ray from left to right, and one is subtracted each time a segment crosses the ray from right to left. If the result equals zero, the point is inside the contour. Otherwise, it is outside it.

#### getLength

```
{Integer} getLength()
```

**Returns** the number of contours in the polygon.

## IPixelRectangleGeometry

Extends [IPixelGeometry](#).

Interface for the "Rectangle" pixel geometry.

[Constructor](#) | [Fields](#) | [Methods](#)

#### Constructor

```
IPixelRectangleGeometry()
```

#### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

**Methods**

Name	Returns	Description
<a href="#">equals(geometry)</a>	Boolean	Returns true if the passed geometry is equivalent to the given one.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for the point nearest to "anchorPosition" on the rectangle.
<a href="#">getCoordinates()</a>	Number[][]	Returns coordinates of two opposite corners of the rectangle.
<a href="#">getMetaData()</a>	Object	Returns metadata of the pixel geometry.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">getType()</a>	String	Returns ID of the geometry type.  Inherited from <a href="#">IBaseGeometry</a> .
<a href="#">scale(factor)</a>	<a href="#">IPixelGeometry</a>	Creates a scaled copy of the geometry.  Inherited from <a href="#">IPixelGeometry</a> .
<a href="#">shift(offset)</a>	<a href="#">IPixelGeometry</a>	Creates a copy of the geometry that is shifted by the specified amount.  Inherited from <a href="#">IPixelGeometry</a> .

**Methods details****getClosest**

```
{Object} getClosest(anchorPosition)
```

Searches for the point nearest to "anchorPosition" on the rectangle.

**Returns** an object with the following fields:

- `position` - The point on the rectangle that is nearest to "anchorPosition".
- `distance` - Distance from "anchorPosition" to "position".
- `closestPointIndex` - Index of the rectangle vertex closest to "position".
- `nextPointIndex` - Index of the rectangle vertex that follows "position".
- `prevPointIndex` - Index of the rectangle vertex that precedes "position".
- `pathIndex` - Index of the rectangle contour that the found point is associated with.

The "nextPointIndex" and "prevPointIndex" fields may be omitted if "position" coincides with one of the rectangle vertexes.

#### Parameters:

Parameter	Default value	Description
<code>anchorPosition</code> *	—	Type: <code>Number[]</code>  Coordinates of a point for which the nearest rectangle vertex is calculated.

\* Mandatory parameter/option.

#### getCoordinates

```
{Number[][]} getCoordinates()
```

**Returns** coordinates of two opposite corners of the rectangle.

## IPointGeometry

Extends [IGeometry](#), [IPointGeometryAccess](#).

Interface for the "Point" geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
IPointGeometry()
```

#### Fields

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .
<code>options</code>	<a href="#">IOptionManager</a>	Options manager.  Inherited from <a href="#">ICustomizable</a> .

#### Events

Name	Description
<code>change</code>	Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>• <code>oldCoordinates</code> - Old coordinates</li> <li>• <code>newCoordinates</code> - New coordinates.</li> </ul> Inherited from <a href="#">IPointGeometryAccess</a> .

Name	Description
<a href="#">mapchange</a>	Map reference changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> Inherited from <a href="#">IGeometry</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">pixelgeometrychange</a>	The pixel geometry changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>pixelGeometry - New <a href="#">IPixelGeometry</a> pixel geometry.</li> </ul> Inherited from <a href="#">IGeometry</a> .

## Methods

Name	Returns	Description
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getCoordinates()</a>	Number[] null	Returns coordinates of a point.  Inherited from <a href="#">IPointGeometryAccess</a> .
<a href="#">getMap()</a>	Map null	Returns the current map.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getPixelGeometry([options])</a>	<a href="#">IPixelGeometry</a>	Returns the pixel geometry corresponding to the given geometry, its options, and the map state.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getType()</a>	String	Returns the "Point" string.
<a href="#">setCoordinates(coordinates)</a>	<a href="#">IPointGeometryAccess</a>	Sets coordinates of a point.  Inherited from <a href="#">IPointGeometryAccess</a> .
<a href="#">setMap(map)</a>		Sets the map.  Inherited from <a href="#">IGeometry</a> .

## Methods details

### getType

```
{String} getType()
```

**Returns** the "Point" string.

## IPointGeometryAccess

Interface for access to the "Point" geometry.

[Constructor](#) | [Events](#) | [Methods](#)

### Constructor

```
IPointGeometryAccess()
```

### Events

Name	Description
<a href="#">change</a>	Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"><li>oldCoordinates - Old coordinates</li><li>newCoordinates - New coordinates.</li></ul>

### Methods

Name	Returns	Description
<a href="#">getCoordinates()</a>	Number[] null	Returns coordinates of a point.
<a href="#">setCoordinates(coordinates)</a>	<a href="#">IPointGeometryAccess</a>	Sets coordinates of a point.

### Events details

#### change

Changed coordinates. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- oldCoordinates - Old coordinates
- newCoordinates - New coordinates.

### Methods details

#### getCoordinates

```
{Number[]|null} getCoordinates()
```

**Returns** coordinates of a point.

#### setCoordinates

```
{IPointGeometryAccess} setCoordinates(coordinates)
```

Sets coordinates of a point.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<a href="#">coordinates</a> *	—	Type: Number[][] null Coordinates of a point.

\* Mandatory parameter/option.

## IPolygonGeometry

Extends [IGeometry](#), [IPolygonGeometryAccess](#).

Interface for the "Polygon" geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IPolygonGeometry()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

### Events

Name	Description
<a href="#">change</a>	Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>oldCoordinates</code> - Old coordinates</li> <li><code>newCoordinates</code> - New coordinates.</li> <li><code>oldFillRule</code> - Old fill rule.</li> <li><code>newFillRule</code> - New fill rule.</li> </ul> Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">mapchange</a>	Map reference changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>oldMap</code> - Old map.</li> <li><code>newMap</code> - New map.</li> </ul> Inherited from <a href="#">IGeometry</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">pixelgeometrychange</a>	The pixel geometry changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>pixelGeometry</code> - New <a href="#">IPixelGeometry</a> pixel geometry.</li> </ul> Inherited from <a href="#">IGeometry</a> .

## Methods

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the polygon.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	Switches the object to "frozen" mode.  Inherited from <a href="#">IFreezable</a> .
<a href="#">get(index)</a>	Number[][]	Returns coordinates of the contour with the specified index.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getChildGeometry(index)</a>	<a href="#">ILinearRingGeometryAccess</a>	Creates and returns an <a href="#">ILinearRingGeometryAccess</a> object for the specified contour.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for the point nearest to "anchorPosition" on the polygon contour.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">getCoordinates()</a>	Number[][][]	Returns an array of geometry coordinates.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">getFillRule()</a>	String	Returns ID of the fill rule.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">getLength()</a>	Integer	Returns the number of contours in the geometry.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">getMap()</a>	<a href="#">Map</a>  null	Returns the current map.  Inherited from <a href="#">IGeometry</a> .

Name	Returns	Description
<a href="#">getPixelGeometry([options])</a>	<a href="#">IPixelGeometry</a>	Returns the pixel geometry corresponding to the given geometry, its options, and the map state.  Inherited from <a href="#">IGeometry</a> .
<a href="#">getType()</a>	String	Returns the "Polygon" string.
<a href="#">insert(index, path)</a>	<a href="#">IPolygonGeometryAccess</a>	Adds a new contour with the specified index.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false.  Inherited from <a href="#">IFreezable</a> .
<a href="#">remove(index)</a>	<a href="#">ILinearRingGeometryAccess</a>	Removes the contour with the specified index.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">set(index, path)</a>	<a href="#">IPolygonGeometryAccess</a>	Sets coordinates of the contour with the specified index.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">setCoordinates(coordinates)</a>	<a href="#">IPolygonGeometryAccess</a>	Sets an array of geometry coordinates.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">setFillRule(fillRule)</a>	<a href="#">IPolygonGeometryAccess</a>	Sets the polygon's fill rule.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">setMap(map)</a>		Sets the map.  Inherited from <a href="#">IGeometry</a> .
<a href="#">splice(index, number)</a>	<a href="#">ILinearRingGeometryAccess[]</a>	Deletes a defined number of contours, starting from the specified index. New contours can be added in place of the deleted ones. Coordinates of the new contours can be passed as additional arguments after the "number" parameter.  Inherited from <a href="#">IPolygonGeometryAccess</a> .
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode.  Inherited from <a href="#">IFreezable</a> .

## Methods details

### getType

```
{String} getType()
```

Returns the "Polygon" string.

## IPolygonGeometryAccess

Extends [IFreezable](#).

Interface for accessing the "Polygon" geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IPolygonGeometryAccess()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager for the object. Inherited from <a href="#">IFreezable</a> .

### Events

Name	Description
<a href="#">change</a>	Changed coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>oldCoordinates</code> - Old coordinates</li> <li><code>newCoordinates</code> - New coordinates.</li> <li><code>oldFillRule</code> - Old fill rule.</li> <li><code>newFillRule</code> - New fill rule.</li> </ul>

### Methods

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the polygon.
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	Switches the object to "frozen" mode. Inherited from <a href="#">IFreezable</a> .
<a href="#">get(index)</a>	Number[][]	Returns coordinates of the contour with the specified index.
<a href="#">getChildGeometry(index)</a>	<a href="#">ILinearRingGeometryAccess</a>	Creates and returns an <a href="#">ILinearRingGeometryAccess</a> object for the specified contour.

Name	Returns	Description
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for the point nearest to "anchorPosition" on the polygon contour.
<a href="#">getCoordinates()</a>	Number[][][]	Returns an array of geometry coordinates.
<a href="#">getFillRule()</a>	String	Returns ID of the fill rule.
<a href="#">getLength()</a>	Integer	Returns the number of contours in the geometry.
<a href="#">insert(index, path)</a>	<a href="#">IPolygonGeometryAccess</a>	Adds a new contour with the specified index.
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false.  Inherited from <a href="#">IFreezable</a> .
<a href="#">remove(index)</a>	<a href="#">ILinearRingGeometryAccess</a>	Removes the contour with the specified index.
<a href="#">set(index, path)</a>	<a href="#">IPolygonGeometryAccess</a>	Sets coordinates of the contour with the specified index.
<a href="#">setCoordinates(coordinates)</a>	<a href="#">IPolygonGeometryAccess</a>	Sets an array of geometry coordinates.
<a href="#">setFillRule(fillRule)</a>	<a href="#">IPolygonGeometryAccess</a>	Sets the polygon's fill rule.
<a href="#">splice(index, number)</a>	<a href="#">ILinearRingGeometryAccess[]</a>	Deletes a defined number of contours, starting from the specified index. New contours can be added in place of the deleted ones. Coordinates of the new contours can be passed as additional arguments after the "number" parameter.
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode.  Inherited from <a href="#">IFreezable</a> .

## Events details

### change

Changed coordinates. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `oldCoordinates` - Old coordinates
- `newCoordinates` - New coordinates.
- `oldFillRule` - Old fill rule.
- `newFillRule` - New fill rule.

## Methods details

### contains

```
{Boolean} contains(position)
```

Checks whether the passed point is located inside the polygon.

**Returns** indicator for whether the point belongs to the polygon.

#### Parameters:

Parameter	Default value	Description
<a href="#">position</a> *	—	Type: Number[] Coordinates of a point.

\* Mandatory parameter/option.

#### Example:

```
var myPolygon = new ymaps.geometry.Polygon([
  [[69, 45], [68, 55], [86, 56]]
]);

// This method only works with a map that is set correctly.
myPolygon.options.setParent(myMap.options);
myPolygon.geometry.setMap(myMap);

// Checking whether the click point is inside the polygon with the geometry set above.
myMap.events.add('click', function (e) {
  alert(myPolygon.geometry.contains(e.get('coords')) ? 'Hit!' : 'Miss!');
});
```

### get

```
{Number[][]} get(index)
```

**Returns** coordinates of the contour with the specified index.

#### Parameters:

Parameter	Default value	Description
<a href="#">index</a> *	—	Type: Integer Contour index.

\* Mandatory parameter/option.

### getChildGeometry

```
{ILinearRingGeometryAccess} getChildGeometry(index)
```

Creates and returns an [ILinearRingGeometryAccess](#) object for the specified contour.

**Returns** the geometry object that corresponds to the specified contour.

#### Parameters:

Parameter	Default value	Description
<a href="#">index</a> *	—	Type: Integer Contour index.

\* Mandatory parameter/option.

### getClosest

```
{Object} getClosest(anchorPosition)
```

Searches for the point nearest to "anchorPosition" on the polygon contour.

**Returns** an object with the following fields:

- position - The point on the polygon contour that is nearest to "anchorPosition".
- distance - Distance from "anchorPosition" to "position".
- closestPointIndex - Index of the polygon vertex closest to "position".
- nextPointIndex - Index of the polygon vertex that follows "position".
- prevPointIndex - Index of the polygon vertex that precedes "position".
- pathIndex - Index of the polygon contour that the found point is associated with.

The "nextPointIndex" and "prevPointIndex" fields may be omitted if "position" coincides with one of the polygon vertexes.

**Parameters:**

Parameter	Default value	Description
<code>anchorPosition</code> *	—	Type: Number[]  Coordinates of a point for which the nearest vertex on the polygon contour is calculated.

\* Mandatory parameter/option.

**Example:**

```
var myPolygon = new ymaps.Polygon([
  [[69, 45], [68, 55], [86, 56], [87, 43]],
  [[60, 48], [62, 60], [80, 62], [80, 48]]
]);
var myPoint = new ymaps.Placemark([74, 52]);
// Constructing the shortest normal from the polygon contour to the point:
var normalVec = new ymaps.Polyline([
  myPolygon.geometry.getClosest(myPoint.geometry.getCoordinates()).position,
  myPoint.geometry.getCoordinates()
]);

myMap.geoObjects
  .add(myPolygon)
  .add(myPoint);
  .add(normalVec);
```

### getCoordinates

```
{Number[][][]} getCoordinates()
```

**Returns** an array of geometry coordinates.

### getFillRule

```
{String} getFillRule()
```

**Returns** ID of the fill rule.

**Example:**

```
var myPolygon = new ymaps.Polygon([
  [[69, 45], [68, 55], [86, 56], [87, 43]],
  [[60, 48], [62, 60], [80, 62], [80, 48]]
]);
```

```

var middlePoint = [74, 52];

myMap.geoObjects.add(myPolygon);

// Checking a point that falls on the intersection of two contours
// by default (with the value fillRule == 'evenOdd') forms an opening (hole).
alert(myPolygon.geometry.contains(middlePoint)); // => false

// After setting the value to 'nonZero', all intersections are also "filled in",
// so now the point is in the polygon.
myPolygon.geometry.setFillRule('nonZero');
alert(myPolygon.geometry.contains(middlePoint)); // => true

```

## getLength

```
{Integer} getLength()
```

**Returns** the number of contours in the geometry.

## insert

```
{IPolygonGeometryAccess} insert(index, path)
```

Adds a new contour with the specified index.

**Returns** self-reference.

### Parameters:

Parameter	Default value	Description
<code>index</code> *	—	Type: Integer Contour index.
<code>path</code> *	—	Type: Number[][] Contour coordinates.

\* Mandatory parameter/option.

## remove

```
{ILinearRingGeometryAccess} remove(index)
```

Removes the contour with the specified index.

**Returns** the deleted contour.

### Parameters:

Parameter	Default value	Description
<code>index</code> *	—	Type: Integer Contour index.

\* Mandatory parameter/option.

## set

```
{IPolygonGeometryAccess} set(index, path)
```

Sets coordinates of the contour with the specified index.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>index</code> *	—	Type: Integer Contour index.
<code>path</code> *	—	Type: Number[][] Contour coordinates.

\* Mandatory parameter/option.

**setCoordinates**

```
{IPolygonGeometryAccess} setCoordinates(coordinates)
```

Sets an array of geometry coordinates.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>coordinates</code> *	—	Type: Number[][][] Geometry coordinates.

\* Mandatory parameter/option.

**setFillRule**

```
{IPolygonGeometryAccess} setFillRule(fillRule)
```

Sets the polygon's fill rule.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>fillRule</code> *	—	Type: String ID of the fill rule.

\* Mandatory parameter/option.

**splice**

```
{ILinearRingGeometryAccess[]} splice(index, number)
```

Deletes a defined number of contours, starting from the specified index. New contours can be added in place of the deleted ones. Coordinates of the new contours can be passed as additional arguments after the "number" parameter.

**Returns** the deleted contours.

**Parameters:**

Parameter	Default value	Description
<a href="#">index</a> *	—	Type: Integer  The index to start from for removing and adding contours.
<a href="#">number</a> *	—	Type: Integer  The number of contours to be deleted.

\* Mandatory parameter/option.

## IPopup

Extends [ICustomizable](#), [IEventEmitter](#).

Interface for an info object.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IPopup ()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

### Events

Name	Description
<a href="#">close</a>	Closing the info object.
<a href="#">open</a>	Opening the info object.
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .

### Methods

Name	Returns	Description
<a href="#">close</a> ( <a href="#">[force]</a> )	<a href="#">vow.Promise</a>	Closes the info object.
<a href="#">getData</a> ()		Returns info object data.
<a href="#">getOverlay</a> ()	<a href="#">vow.Promise</a>	Returns the promise object to return the overlay.

Name	Returns	Description
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>	Returns the overlay, if one exists.
<a href="#">getPosition()</a>		Returns the coordinates of the info object.
<a href="#">isOpen()</a>	Boolean	Returns the info object state: open/closed.
<a href="#">open([position[, data]])</a>	<a href="#">vow.Promise</a>	Opens the info object at the specified position. If the info object is already open, it moves it to the specified point. The format and content of the coordinates is determined by the <a href="#">IProjection</a> that is in the options.
<a href="#">setData(data)</a>	<a href="#">vow.Promise</a>	Defines new data for the info object.
<a href="#">setPosition(position)</a>	<a href="#">vow.Promise</a>	Specifies a new position for the info object.

### Events details

#### close

Closing the info object.

#### open

Opening the info object.

### Methods details

#### close

```
{vow.Promise} close([force])
```

Closes the info object.

**Returns** Promise object.

#### Parameters:

Parameter	Default value	Description
<a href="#">force</a>	false	Type: Boolean Instant closure.

#### getData

```
{ } getData()
```

**Returns** info object data.

**getOverlay**

```
{vow.Promise} getOverlay()
```

**Returns** the promise object to return the overlay.

**getOverlaySync**

```
{IOverlay} getOverlaySync()
```

**Returns** the overlay, if one exists.

**getPosition**

```
{ } getPosition()
```

**Returns** the coordinates of the info object.

**isOpen**

```
{Boolean} isOpen()
```

**Returns** the info object state: open/closed.

**open**

```
{vow.Promise} open([position[, data]])
```

Opens the info object at the specified position. If the info object is already open, it moves it to the specified point. The format and content of the coordinates is determined by the [IProjection](#) that is in the options.

**Returns** Promise object.

**Parameters:**

Parameter	Default value	Description
<a href="#">position</a>	—	Type: Number[]  The point where you want to place the balloon.
<a href="#">data</a>	—	Type: Object String HTMLElement  Overlay data.

**setData**

```
{vow.Promise} setData(data)
```

Defines new data for the info object.

**Returns** Promise object.

**Parameters:**

Parameter	Default value	Description
<a href="#">data</a> *	—	Type: Object String HTMLElement  Info object data.

\* Mandatory parameter/option.

### setPosition

```
{vow.Promise} setPosition(position)
```

Specifies a new position for the info object.

**Returns** Promise object.

#### Parameters:

Parameter	Default value	Description
<code>position</code> *	—	Type: Number[] The coordinates of the info object.

\* Mandatory parameter/option.

## IPopupManager

Extends [IEventEmitter](#).

Interface for the info object manager.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IPopupManager()
```

### Fields

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

### Events

Name	Description
<code>close</code>	Closing the info object. Names of fields available via <a href="#">Event.get</a> : <ul style="list-style-type: none"> <li><code>target</code> - Reference to the object where the closing occurred.</li> </ul>
<code>open</code>	Opening the info object. Names of fields available via <a href="#">Event.get</a> : <ul style="list-style-type: none"> <li><code>target</code> - Reference to the object where the opening occurred.</li> </ul>

### Methods

Name	Returns	Description
<code>close([force])</code>	<a href="#">vow.Promise</a>	Closes the info object.
<code>destroy()</code>		Disables the info object manager.
<code>getData()</code>	Object null	Returns the data of the info object or 'null'.

Name	Returns	Description
<code>getOptions()</code>	<code>IOptionManager</code>  null	Returns the options manager or 'null'.
<code>getOverlay()</code>	<code>vow.Promise</code>	Returns the promise object to return the overlay.
<code>getOverlaySync()</code>	<code>IOverlay</code>  null	Returns the overlay, if one exists.
<code>getPosition()</code>	<code>Number[]</code>  null	Returns the coordinates of the info object or 'null'.
<code>isOpen()</code>	<code>Boolean</code>	Returns the info object state: open/closed.
<code>open([position[, data[, options]])</code>	<code>vow.Promise</code>	Opens the info object at the specified position.
<code>setData(data)</code>	<code>vow.Promise</code>	Defines new data for the info object.
<code>setOptions(options)</code>	<code>vow.Promise</code>	Defines new options for the info object.
<code>setPosition(position)</code>	<code>vow.Promise</code>	Specifies a new position for the info object.

### Events details

#### close

Closing the info object. Names of fields available via [Event.get](#):

- target - Reference to the object where the closing occurred.

#### open

Opening the info object. Names of fields available via [Event.get](#):

- target - Reference to the object where the opening occurred.

### Methods details

#### close

```
{vow.Promise} close([force])
```

Closes the info object.

**Returns** Promise object.

#### Parameters:

Parameter	Default value	Description
<code>force</code>	false	Type: Boolean Instant closure.

#### destroy

```
{ } destroy()
```

Disables the info object manager.

### getData

```
{Object|null} getData()
```

Returns the data of the info object or 'null'.

### getOptions

```
{IOptionManager|null} getOptions()
```

Returns the options manager or 'null'.

### getOverlay

```
{vow.Promise} getOverlay()
```

Returns the promise object to return the overlay.

### getOverlaySync

```
{IOverlay|null} getOverlaySync()
```

Returns the overlay, if one exists.

### getPosition

```
{Number[]|null} getPosition()
```

Returns the coordinates of the info object or 'null'.

### isOpen

```
{Boolean} isOpen()
```

Returns the info object state: open/closed.

### open

```
{vow.Promise} open([position[, data[, options]])
```

Opens the info object at the specified position.

Returns Promise object.

#### Parameters:

Parameter	Default value	Description
<a href="#">position</a>	—	Type: Number[]  Coordinates of the point where the hint is opened.
<a href="#">data</a>	—	Type: Object String HTML Element  Data.

Parameter	Default value	Description
<code>options</code>	—	Type: Object Options.

**setData**

```
{vow.Promise} setData(data)
```

Defines new data for the info object.

**Returns** Promise object.

**Parameters:**

Parameter	Default value	Description
<code>data *</code>	—	Type: Object String  HTMLElement Info object data.

\* Mandatory parameter/option.

**setOptions**

```
{vow.Promise} setOptions(options)
```

Defines new options for the info object.

**Returns** Promise object.

**Parameters:**

Parameter	Default value	Description
<code>options *</code>	—	Type: Object Info object options.

\* Mandatory parameter/option.

**setPosition**

```
{vow.Promise} setPosition(position)
```

Specifies a new position for the info object.

**Returns** Promise object.

**Parameters:**

Parameter	Default value	Description
<code>position *</code>	—	Type: Number[] The coordinates of the info object.

\* Mandatory parameter/option.

## IPositioningContext

Interface for the positioning context, an object that allows to position an object inside itself that is defined by global pixel coordinates.

[Constructor](#) | [Methods](#)

### Constructor

```
IPositioningContext()
```

### Methods

Name	Returns	Description
<a href="#">fromClientPixels(clientPixelPoint)</a>	Number[]	Converts client pixel coordinates to global coordinates.
<a href="#">getZoom()</a>	Number	Returns the current zoom level at which the positioning context works.
<a href="#">toClientPixels(globalPixelPoint)</a>	Number[]	Converts global pixel coordinates to client coordinates.

### Methods details

#### fromClientPixels

```
{Number[]} fromClientPixels(clientPixelPoint)
```

Converts client pixel coordinates to global coordinates.

**Returns** global pixel coordinates.

#### Parameters:

Parameter	Default value	Description
<a href="#">clientPixelPoint</a> *	—	Type: Number[] Client pixel coordinates.

\* Mandatory parameter/option.

#### getZoom

```
{Number} getZoom()
```

**Returns** the current zoom level at which the positioning context works.

#### toClientPixels

```
{Number[]} toClientPixels(globalPixelPoint)
```

Converts global pixel coordinates to client coordinates.

**Returns** client pixel coordinates.

#### Parameters:

Parameter	Default value	Description
<a href="#">globalPixelPoint</a> *	—	Type: Number[] Global pixel coordinates.

\* Mandatory parameter/option.

## IProjection

Projection. Describes how the real map is projected onto the endless pixel plane. One "world" should be sized 256x256 pixels at the zero zoom level, while the upper left corner of this world has the coordinates (0,0) and the axes coordinates go to the right and down. "Worlds" can be connected along any axis (or both axes at once).

[Constructor](#) | [Methods](#)

### Constructor

```
IProjection()
```

### Methods

Name	Returns	Description
<a href="#">fromGlobalPixels(globalPixelPoint, zoom)</a>	Number[]	Converts pixel coordinates to the projection's coordinates at the specified zoom level.
<a href="#">getCoordSystem()</a>	<a href="#">ICoordSystem</a>	Returns the coordinate system that is used by the projection.
<a href="#">isCycled()</a>	Boolean[]	Indicator of projection cycling.
<a href="#">toGlobalPixels(coordPoint, zoom)</a>	Number[]	Converts projection coordinates to global pixel coordinates at the specified zoom level.

### Methods details

#### fromGlobalPixels

```
{Number[]} fromGlobalPixels(globalPixelPoint, zoom)
```

Converts pixel coordinates to the projection's coordinates at the specified zoom level.

**Returns** a point in the projection's coordinates.

#### Parameters:

Parameter	Default value	Description
<a href="#">globalPixelPoint</a> *	—	Type: Number[] A point in pixel coordinates.
<a href="#">zoom</a> *	—	Type: Number Zoom level.

\* Mandatory parameter/option.

## getCoordSystem

```
{ICoordSystem} getCoordSystem()
```

**Returns** the coordinate system that is used by the projection.

## isCycled

```
{Boolean[]} isCycled()
```

Indicator of projection cycling.

**Returns** a pair of flags that show whether the map is looped along the pixel axes (x/y).

## toGlobalPixels

```
{Number[]} toGlobalPixels(coordPoint, zoom)
```

Converts projection coordinates to global pixel coordinates at the specified zoom level.

**Returns** a pair of pixel coordinates.

### Parameters:

Parameter	Default value	Description
<code>coordPoint</code> *	—	Type: Number[]  A point in the projection's coordinates.
<code>zoom</code> *	—	Type: Number  Zoom level.

\* Mandatory parameter/option.

## IPromiseProvider

Object fulfilling the promise.

[Constructor](#) | [Methods](#)

### Constructor

```
IPromiseProvider()
```

### Methods

Name	Returns	Description
<code>then(onResolve, onReject)</code>	<a href="#">IPromiseProvider</a>	Returns a self-reference or a new Promise object.

### Methods details

#### then

```
{IPromiseProvider} then(onResolve, onReject)
```

**Returns** a self-reference or a new Promise object.

### Parameters:

Parameter	Default value	Description
<a href="#">onResolve</a> *	—	Type: Function  Handler function that is called if the promise was fulfilled.
<a href="#">onReject</a> *	—	Type: Function  Handler function that is called if the promise was not fulfilled (an error occurred).

\* Mandatory parameter/option.

## IRatioMap

Interface of an object containing the ratio of custom data to devicePixelRatio. Used when needed to support screens with a ratio of virtual to physical pixels greater than one. As keys for the object, use strings consisting of whole numbers or fractions indicating the pixel density coefficient.

### Constructor

#### Constructor

```
IRatioMap()
```

#### Example:

```
// Let's say we need to display the image correctly.
var images = {
  // For regular screens we'll display the normal picture.
  "1": "100x100.png",
  // For HTC Desire, Samsung Galaxy S II and others with devicePixelRatio = 1.5.
  "1.5": "150x150.png",
  // For Apple devices with the Retina screen, as well as for Sony Xperia S, HTC One X and others
  // with the ratio = 2.
  "2": "200x100.png"
}
```

## IRectangleGeometry

Extends [IGeometry](#), [IRectangleGeometryAccess](#).

Interface for the "Rectangle" geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
IRectangleGeometry()
```

#### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager.  Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">change</a>	<p>Change to corner coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>oldCoordinates - Old coordinates of the corners.</li> <li>newCoordinates - New coordinates of the corners.</li> </ul> <p>Inherited from <a href="#">IRectangleGeometryAccess</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> <p>Inherited from <a href="#">IGeometry</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">pixelgeometrychange</a>	<p>The pixel geometry changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>pixelGeometry - New <a href="#">IPixelGeometry</a> pixel geometry.</li> </ul> <p>Inherited from <a href="#">IGeometry</a>.</p>

## Methods

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	<p>Checks whether the passed point is located inside the rectangle.</p> <p>Inherited from <a href="#">IRectangleGeometryAccess</a>.</p>
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	<p>Switches the object to "frozen" mode.</p> <p>Inherited from <a href="#">IFreezable</a>.</p>
<a href="#">getBounds()</a>	Number[][] null	<p>Returns coordinates of the two opposite corners of the area that surrounds the geometry. The first item in the array is the southwest corner of the area; the second item is the northeast corner.</p> <p>Inherited from <a href="#">IGeometry</a>.</p>
<a href="#">getClosest(anchorPosition)</a>	Object	<p>Searches for the point nearest to "anchorPosition" on the rectangle contour.</p> <p>Inherited from <a href="#">IRectangleGeometryAccess</a>.</p>
<a href="#">getCoordinates()</a>	Number[][]	<p>Returns coordinates of two opposite corners of the rectangle.</p> <p>Inherited from <a href="#">IRectangleGeometryAccess</a>.</p>

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>  null	Returns the current map. Inherited from <a href="#">IGeometry</a> .
<a href="#">getPixelGeometry([options])</a>	<a href="#">IPixelGeometry</a>	Returns the pixel geometry corresponding to the given geometry, its options, and the map state. Inherited from <a href="#">IGeometry</a> .
<a href="#">getType()</a>	String	Returns the "Rectangle" string.
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false. Inherited from <a href="#">IFreezable</a> .
<a href="#">setCoordinates(coordinates)</a>	<a href="#">IRectangleGeometryAccess</a>	Sets the coordinates of two opposite corners of the rectangle. Inherited from <a href="#">IRectangleGeometryAccess</a> .
<a href="#">setMap(map)</a>		Sets the map. Inherited from <a href="#">IGeometry</a> .
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	Switches the object to active mode. Inherited from <a href="#">IFreezable</a> .

## Methods details

### getType

```
{String} getType()
```

Returns the "Rectangle" string.

## IRectangleGeometryAccess

Extends [IFreezable](#).

Interface for accessing the "Rectangle" geometry.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IRectangleGeometryAccess()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager for the object. Inherited from <a href="#">IFreezable</a> .

## Events

Name	Description
<a href="#">change</a>	<p>Change to corner coordinates. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>oldCoordinates - Old coordinates of the corners.</li> <li>newCoordinates - New coordinates of the corners.</li> </ul>

## Methods

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the rectangle.
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	<p>Switches the object to "frozen" mode.</p> <p>Inherited from <a href="#">IFreezable</a>.</p>
<a href="#">getClosest(anchorPosition)</a>	Object	Searches for the point nearest to "anchorPosition" on the rectangle contour.
<a href="#">getCoordinates()</a>	Number[][]	Returns coordinates of two opposite corners of the rectangle.
<a href="#">isFrozen()</a>	Boolean	<p>Returns true if the object is in "frozen" mode, otherwise false.</p> <p>Inherited from <a href="#">IFreezable</a>.</p>
<a href="#">setCoordinates(coordinates)</a>	<a href="#">IRectangleGeometryAccess</a>	Sets the coordinates of two opposite corners of the rectangle.
<a href="#">unfreeze()</a>	<a href="#">IFreezable</a>	<p>Switches the object to active mode.</p> <p>Inherited from <a href="#">IFreezable</a>.</p>

## Events details

### change

Change to corner coordinates. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- oldCoordinates - Old coordinates of the corners.
- newCoordinates - New coordinates of the corners.

## Methods details

### contains

```
{Boolean} contains(position)
```

Checks whether the passed point is located inside the rectangle.

**Returns** indicator for whether the point belongs to the rectangle.

**Parameters:**

Parameter	Default value	Description
<a href="#">position</a> *	—	Type: Number[] Coordinates of a point.

\* Mandatory parameter/option.

**getClosest**

```
{Object} getClosest(anchorPosition)
```

Searches for the point nearest to "anchorPosition" on the rectangle contour.

**Returns** an object with the following fields:

- `position` - The point on the rectangle contour that is nearest to "anchorPosition".
- `distance` - Distance from "anchorPosition" to "position".

**Parameters:**

Parameter	Default value	Description
<a href="#">anchorPosition</a> *	—	Type: Number[] Coordinates of a point for which the nearest rectangle vertex is calculated.

\* Mandatory parameter/option.

**getCoordinates**

```
{Number[][]} getCoordinates()
```

**Returns** coordinates of two opposite corners of the rectangle.

**setCoordinates**

```
{IRectangleGeometryAccess} setCoordinates(coordinates)
```

Sets the coordinates of two opposite corners of the rectangle.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<a href="#">coordinates</a> *	—	Type: Number[][] Coordinates of corners.

\* Mandatory parameter/option.

**IRoutePanel**

Extends [IEventEmitter](#).

Interface for route panel.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

IRoutePanel()

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Option manager. Names of options: <ul style="list-style-type: none"> <li><code>allowSwitch</code>: Boolean = true - Whether button for switching way points should be shown.</li> <li><code>reverseGeocoding</code>: Boolean = true - Whether reverse geocoding should be enabled during routing.</li> <li><code>adjustMapMargin</code>: Boolean = false - Whether the panel registers its size in the map margins manager <a href="#">map.margin.Manager</a>.</li> <li><code>types</code>: Object = { auto: true, masstransit: true, pedestrian: true, taxi: false } - Specifies routing modes available for user to select in routing panel. When set, <code>state.type</code> is automatically adjusted, if current <code>state.type</code> becomes unavailable. Types are shown in panel only if two or more are available for user to select.</li> </ul>
<a href="#">state</a>	<a href="#">IDataManager</a>	State manager. Names of states: <ul style="list-style-type: none"> <li><code>type</code>: String - Routing type <a href="#">IMultiRouteParams.routingMode</a>.</li> <li><code>fromEnabled</code>: Boolean - Enables the "from" field for users to enter the route origin.</li> <li><code>from</code>: String - Address or coordinates of departure.</li> <li><code>toEnabled</code>: Boolean - Enables the "to" field for users to enter the route destination.</li> <li><code>to</code>: String - Address or coordinates of arrival.</li> </ul>

**Events**

Name	Description
<a href="#">disable</a>	Route panel is unloaded.
<a href="#">enable</a>	Route panel and it's dependencies are loaded and ready for user interactions.

**Methods**

Name	Returns	Description
<a href="#">enable()</a>		Loads panel dependencies and enables it for usage.
<a href="#">geolocate(name)</a>	<a href="#">vow.Promise</a>	Use user's geolocation as coordinates for 'from' or 'to'.
<a href="#">getRoute()</a>	<a href="#">multiRouter.MultiRoute</a>	 <b>Attention:</b> This method is deprecated. See <a href="#">IRoutePanel.getRouteAsync</a> . Returns built route.

Name	Returns	Description
<a href="#">getRouteAsync()</a>	<a href="#">vow.Promise</a> .< <a href="#">multiRouter.MultiRoute</a> >	Returns <a href="#">vow.Promise</a> , which will be resolved with built route. If an error occurs, the promise object is rejected.
<a href="#">isEnabled()</a>	Boolean	Returns whether panel is fully loaded.
<a href="#">switchPoints()</a>		Switches points (and corresponding inputs).

## Fields details

### options

```
{IOptionManager} options
```

Option manager. Names of options:

- `allowSwitch`: Boolean = true - Whether button for switching way points should be shown.
- `reverseGeocoding`: Boolean = true - Whether reverse geocoding should be enabled during routing.
- `adjustMapMargin`: Boolean = false - Whether the panel registers its size in the map margins manager [map.margin.Manager](#).
- `types`: Object = { `auto`: true, `masstransit`: true, `pedestrian`: true, `taxi`: false } - Specifies routing modes available for user to select in routing panel. When set, `state.type` is automatically adjusted, if current `state.type` becomes unavailable. Types are shown in panel only if two or more are available for user to select.

### state

```
{IDataManager} state
```

State manager. Names of states:

- `type`: String - Routing type [IMultiRouteParams.routingMode](#).
- `fromEnabled`: Boolean - Enables the "from" field for users to enter the route origin.
- `from`: String - Address or coordinates of departure.
- `toEnabled`: Boolean - Enables the "to" field for users to enter the route destination.
- `to`: String - Address or coordinates of arrival.

## Events details

### disable

Route panel is unloaded.

### enable

Route panel and it's dependencies are loaded and ready for user interactions.

## Methods details

### enable

```
{ } enable ()
```

Loads panel dependencies and enables it for usage.

## geolocate

```
{vow.Promise} geolocate(name)
```

Use user's geolocation as coordinates for 'from' or 'to'.

**Returns** Promise object. See [geolocation](#).

**Parameters:**

Parameter	Default value	Description
<code>name</code> *	—	Type: String  Input to geolocate. Either 'from' or 'to'.

\* Mandatory parameter/option.

## getRoute

```
{multiRouter.MultiRoute} getRoute()
```

**This method is deprecated.** See [IRoutePanel.getRouteAsync](#).

**Returns** built route.

## getRouteAsync

```
{vow.Promise.<multiRouter.MultiRoute>} getRouteAsync()
```

Returns [vow.Promise](#) which:

- will be **resolved** with: `<multiRouter.MultiRoute>` — built route;
- either **rejected** with an error.

## isEnabled

```
{Boolean} isEnabled()
```

**Returns** whether panel is fully loaded.

## switchPoints

```
{ } switchPoints()
```

Switches points (and corresponding inputs).

## ISearchControlLayout

Extends [IExpandableControlLayout](#).

Interface for the layout of the "Search on map" control.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
ISearchControlLayout()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .

**Events**

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">collapse</a>	Event that initiates collapsing an object. Inherited from <a href="#">IExpandableControllLayout</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">emptinesschange</a>	Change to the empty layout indicator. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">ILayout</a> .
<a href="#">expand</a>	Event that initiates expanding an object. Inherited from <a href="#">IExpandableControllLayout</a> .
<a href="#">loadmore</a>	The event that triggers pulling up additional search results. Instance of the <a href="#">Event</a> class.
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseleave</a>	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mousemove</a>	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">parentelementchange</a>	<p>Change to the parent element. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">resultselect</a>	<p>Event that initiates showing the search results on the map. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>index</code> - Number of the object in the server response to show on the map.</li> </ul>
<a href="#">search</a>	<p>Event that initiates searching for objects. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>request</code> - String containing the request.</li> </ul>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the layout. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">destroy()</a>		Destructor. Called when activity with the layout is finished.  Inherited from <a href="#">ILayout</a> .
<a href="#">getData()</a>	Object	Returns layout data object.  Inherited from <a href="#">ILayout</a> .
<a href="#">getParentElement()</a>	HTMLElement	Returns parent HTML element.  Inherited from <a href="#">ILayout</a> .
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element.  Inherited from <a href="#">ILayout</a> .
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.  Inherited from <a href="#">ILayout</a> .
<a href="#">setData(data)</a>		Sets layout data.  Inherited from <a href="#">ILayout</a> .
<a href="#">setParentElement(parent)</a>		Adds the layout to the DOM tree.  Inherited from <a href="#">ILayout</a> .

**Events details****loadmore**

The event that triggers pulling up additional search results. Instance of the [Event](#) class.

**resultselect**

Event that initiates showing the search results on the map. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- index - Number of the object in the server response to show on the map.

**search**

Event that initiates searching for objects. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- request - String containing the request.

## ISearchProvider

Search provider interface.

[Constructor](#) | [Methods](#)

### Constructor

```
ISearchProvider()
```

### Methods

Name	Returns	Description
<a href="#">search</a> (request[, options])	<a href="#">vow.Promise</a>	Sends a request to search for geo objects. A handler function for processing geocoding results can be added via the returned promise object. The object input to the handler function can contain only the following types of fields: geoObjects, layers, mapState, and metaData.
<a href="#">suggest</a> (request[, options])	<a href="#">vow.Promise</a>	Sends a request for search suggestions. Returns a promise object that is either rejected with an error, or confirmed by an array of objects in the format { displayName: "Mitishi, Moscow region", value: "Russia, Moscow region, Mitishi " }. The displayName field represents the toponym in a user-friendly way, and the value field represents the value which should be inserted into the search field after the user selects the suggestion.  This method is optional.

### Methods details

#### search

```
{vow.Promise} search(request[, options])
```

Sends a request to search for geo objects. A handler function for processing geocoding results can be added via the returned promise object. The object input to the handler function can contain only the following types of fields: geoObjects, layers, mapState, and metaData.

**Returns** Promise object.

**Parameters:**

Parameter	Default value	Description
<code>request *</code>	—	Type: String Request string.
<code>options</code>	—	Type: Object Options.
<code>options.boundedBy</code>	—	Type: Number[][] A rectangular area on the map, where the object being searched for is presumably located. Must be set as an array, such as <code>[[30, 40], [50, 50]]</code> .
<code>options.results</code>	—	Type: Number Maximum number of results to be returned.
<code>options.skip</code>	—	Type: Number Number of results that must be skipped.

\* Mandatory parameter/option.

## suggest

```
{vow.Promise} suggest(request[, options])
```

Sends a request for search suggestions. Returns a promise object that is either rejected with an error, or confirmed by an array of objects in the format `{ displayName: "Mitishi, Moscow region", value: "Russia, Moscow region, Mitishi" }`. The `displayName` field represents the toponym in a user-friendly way, and the `value` field represents the value which should be inserted into the search field after the user selects the suggestion.

This method is optional.

**Returns** a Promise object.

### Parameters:

Parameter	Default value	Description
<code>request *</code>	—	Type: String Request string.
<code>options</code>	—	Type: Object Options.
<code>options.boundedBy</code>	—	Type: Number[][] A rectangular area on the map, where the object being searched for is presumably located. Must be set as an array, such as <code>[[30, 40], [50, 50]]</code> .

Parameter	Default value	Description
<a href="#">options.results</a>	—	Type: Number  Maximum number of results to be returned.
<a href="#">options.strictBounds</a>	—	Type: Boolean  Search only inside the area defined by the "boundedBy" option.

\* Mandatory parameter/option.

## ISelectableControl

Extends [IControl](#).

Interface for a control that can be toggled and selected.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
ISelectableControl()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">IControl</a> .

### Events

Name	Description
<a href="#">deselect</a>	The control is not selected.
<a href="#">disable</a>	The control is unavailable.
<a href="#">enable</a>	The control is available.
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li><code>oldParent</code> - Old parent.</li> <li><code>newParent</code> - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .
<a href="#">select</a>	The control is selected.

**Methods**

Name	Returns	Description
<a href="#">deselect()</a>		Cancels selection of the control (turns it off).
<a href="#">disable()</a>		Makes the control unavailable (user actions are not allowed).
<a href="#">enable()</a>		Makes the control available (user actions are allowed).
<a href="#">getParent()</a>	<a href="#">IControlParent</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IControl</a> .
<a href="#">isEnabled()</a>	Boolean	Returns true if the control is available, or false if it is unavailable.
<a href="#">isSelected()</a>	Boolean	Returns true if the control is selected, or false if it is not selected.
<a href="#">select()</a>		Selects (turns on) the control.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IControl</a> .

**Events details****deselect**

The control is not selected.

**disable**

The control is unavailable.

**enable**

The control is available.

**select**

The control is selected.

**Methods details****deselect**

```
{ } deselect()
```

Cancels selection of the control (turns it off).

**disable**

```
{ } disable()
```

Makes the control unavailable (user actions are not allowed).

**enable**

```
{ } enable()
```

Makes the control available (user actions are allowed).

**isEnabled**

```
{Boolean} isEnabled()
```

Returns true if the control is available, or false if it is unavailable.

**isSelected**

```
{Boolean} isSelected()
```

Returns true if the control is selected, or false if it is not selected.

**select**

```
{ } select()
```

Selects (turns on) the control.

**ISelectableControlLayout**

Extends [ILayout](#).

Interface for the button layout.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
ISelectableControlLayout()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .

**Events**

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">deselect</a>	<p>Event that initiates removing an item from the "selected" state.</p>
<a href="#">emptinesschange</a>	<p>Change to the empty layout indicator. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">parentelementchange</a>	<p>Change to the parent element. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">select</a>	<p>Event that initiates putting an item in the "selected" state.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the layout. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		<p>Destructor. Called when activity with the layout is finished.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">getData()</a>	Object	<p>Returns layout data object.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">getParentElement()</a>	HTMLElement	<p>Returns parent HTML element.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	<p>Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element.</p> <p>Inherited from <a href="#">ILayout</a>.</p>

Name	Returns	Description
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.  Inherited from <a href="#">ILayout</a> .
<a href="#">setData(data)</a>		Sets layout data.  Inherited from <a href="#">ILayout</a> .
<a href="#">setParentElement(parent)</a>		Adds the layout to the DOM tree.  Inherited from <a href="#">ILayout</a> .

### Events details

#### deselect

Event that initiates removing an item from the "selected" state.

#### select

Event that initiates putting an item in the "selected" state.

## IShape

Interface of a shape. A shape represents a collection of pixel geometry as well as mathematical and logical parameters for rendering it (such as the presence of a contour and its width, filling, etc.)

[Constructor](#) | [Methods](#)

### Constructor

```
IShape ()
```

### Methods

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the shape.
<a href="#">equals(shape)</a>	Boolean	Returns true if the passed shape is equivalent to the given one.
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area spanning the shape. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.

Name	Returns	Description
<code>getGeometry()</code>	<code>IPixelGeometry</code>	Returns pixel geometry of a shape.
<code>getType()</code>	String	Returns ID of the shape type.
<code>scale(factor)</code>	<code>IShape</code>	Creates a scaled copy of the shape.
<code>shift(offset)</code>	<code>IShape</code>	Creates a copy of the shape that is shifted by the specified amount.

## Methods details

### contains

```
{Boolean} contains(position)
```

Checks whether the passed point is located inside the shape.

**Returns** true if the passed point is inside the shape.

#### Parameters:

Parameter	Default value	Description
<code>position *</code>	—	Type: <code>Number[]</code> Coordinates of a point.

\* Mandatory parameter/option.

### equals

```
{Boolean} equals(shape)
```

**Returns** true if the passed shape is equivalent to the given one.

#### Parameters:

Parameter	Default value	Description
<code>shape *</code>	—	Type: <code>IShape</code> The shape to compare.

\* Mandatory parameter/option.

### getBounds

```
{Number[][]|null} getBounds()
```

**Returns** coordinates of the two opposite corners of the area spanning the shape. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.

### getGeometry

```
{IPixelGeometry} getGeometry()
```

**Returns** pixel geometry of a shape.

### getType

```
{String} getType()
```

**Returns** ID of the shape type.

### scale

```
{IShape} scale(factor)
```

Creates a scaled copy of the shape.

**Returns** a scaled copy of the shape.

**Parameters:**

Parameter	Default value	Description
<code>factor</code> *	—	Type: Number Scaling factor.

\* Mandatory parameter/option.

### shift

```
{IShape} shift(offset)
```

Creates a copy of the shape that is shifted by the specified amount.

**Returns** the shifted copy of the shape.

**Parameters:**

Parameter	Default value	Description
<code>offset</code> *	—	Type: Number[] Amount to shift on the axes.

\* Mandatory parameter/option.

## ISuggestProvider

Interface for a provider of search suggestions.

[Constructor](#) | [Methods](#)

### Constructor

```
ISuggestProvider()
```

## Methods

Name	Returns	Description
<code>suggest(request[, options])</code>	<code>vow.Promise</code>	Sends a request for search suggestions. Returns a promise object that is either rejected with an error, or confirmed by an array of objects in the format { displayName: "Mitishi, Moscow region", value: "Russia, Moscow region, Mitishi", hl: [[0,5]] }. The displayName field is responsible for representing a toponym in a user-friendly format. The value field is the value that must be inserted in the data entry field after the user selects this suggestion. The hl field is an array of ranges for highlighting to show which part of the result matched the query. The range for highlighting is an array of two numbers: the indexes of the starting and ending symbols of the range.

## Methods details

## suggest

```
{vow.Promise} suggest(request[, options])
```

Sends a request for search suggestions. Returns a promise object that is either rejected with an error, or confirmed by an array of objects in the format { displayName: "Mitishi, Moscow region", value: "Russia, Moscow region, Mitishi", hl: [[0,5]] }. The displayName field is responsible for representing a toponym in a user-friendly format. The value field is the value that must be inserted in the data entry field after the user selects this suggestion. The hl field is an array of ranges for highlighting to show which part of the result matched the query. The range for highlighting is an array of two numbers: the indexes of the starting and ending symbols of the range.

**Returns** a Promise object.

**Parameters:**

Parameter	Default value	Description
<code>request</code> *	—	Type: String Request string.
<code>options</code>	—	Type: Object Options.

Parameter	Default value	Description
<a href="#">options.boundedBy</a>	—	Type: Number[][]  A rectangular area on the map, where the object being searched for is presumably located. Must be set as an array, such as [[30, 40], [50, 50]].
<a href="#">options.results</a>	—	Type: Number  Maximum number of results to be returned.

\* Mandatory parameter/option.

## ISuggestViewLayout

Interface for the layout of the search suggestions panel.

[Constructor](#) | [Events](#)

### Constructor

```
ISuggestViewLayout()
```

### Events

Name	Description
<a href="#">hover</a>	Event that occurs when the user selects a suggestion, either by hovering over the result with the mouse or using the keyboard. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>index: Number null - Index of an element in the list of search suggestions.</li> </ul>
<a href="#">select</a>	Event that occurs when the user selects a suggestion. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>item: Object - Object with the "displayName" and "value" fields.</li> </ul>

### Events details

#### hover

Event that occurs when the user selects a suggestion, either by hovering over the result with the mouse or using the keyboard. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- index: Number|null - Index of an element in the list of search suggestions.

#### select

Event that occurs when the user selects a suggestion. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- item: Object - Object with the "displayName" and "value" fields.

## ITile

Extends [IEventEmitter](#), [ITileInfoProvider](#).

Interface for tiles.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

## Constructor

```
ITile(url)
```

### Parameters:

Parameter	Default value	Description
<a href="#">url</a> *	—	Type: String Tile URL.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

## Events

Name	Description
<a href="#">ready</a>	Tile ready event.

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destroys the tile.
<a href="#">isReady()</a>	Boolean	Checks tile readiness.

### Events details

#### ready

Tile ready event.

### Methods details

#### destroy

```
{ } destroy()
```

Destroys the tile.

#### isReady

```
{ Boolean } isReady()
```

Checks tile readiness.

**Returns** true if the tile is ready, or false if it is not ready.

## ITrafficControlLayout

Extends [IExpandableControlLayout](#).

Interface for the layout of the "Traffic panel" control.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
ITrafficControlLayout ()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .

### Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">collapse</a>	Event that initiates collapsing an object. Inherited from <a href="#">IExpandableControlLayout</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">emptinesschange</a>	Change to the empty layout indicator. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">ILayout</a> .
<a href="#">expand</a>	Event that initiates expanding an object. Inherited from <a href="#">IExpandableControlLayout</a> .
<a href="#">hide</a>	Event that initiates deleting traffic from the map. Hide traffic.
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">parentelementchange</a>	<p>Change to the parent element. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">providerkeychange</a>	<p>Event that initiates changing the provider key. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>newProviderKey</code> - New value for the provider key.</li><li>• <code>oldProviderKey</code> - Old key value.</li></ul>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the layout. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>

Name	Description
<a href="#">show</a>	Event that initiates showing traffic on the map.
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		<p>Destructor. Called when activity with the layout is finished.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">getData()</a>	Object	<p>Returns layout data object.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">getParentElement()</a>	HTMLElement	<p>Returns parent HTML element.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	<p>Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">isEmpty()</a>	Boolean	<p>Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">setData(data)</a>		<p>Sets layout data.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">setParentElement(parent)</a>		<p>Adds the layout to the DOM tree.</p> <p>Inherited from <a href="#">ILayout</a>.</p>

## Events details

### hide

Event that initiates deleting traffic from the map. Hide traffic.

**providerkeychange**

Event that initiates changing the provider key. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `newProviderKey` - New value for the provider key.
- `oldProviderKey` - Old key value.

**show**

Event that initiates showing traffic on the map.

**ITrafficProvider**

Extends [ICustomizable](#), [IEventEmitter](#).

Interface for a provider of traffic data.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
ITrafficProvider()
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

**Events**

Name	Description
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .

**Methods**

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>  null	Returns reference to the map.
<a href="#">setMap(Reference)</a>		

**Methods details****getMap**

```
{Map|null} getMap()
```

**Returns** reference to the map.

**setMap**

```
{ } setMap(Reference)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">Reference</a> *	—	Type: <a href="#">Map</a>  null to the map.

\* Mandatory parameter/option.

**ITransportProperties**

This class does not have a constructor and is intended for describing the data object of the vehicle in the transport segment of a public transport route.

[Constructor](#) | [Fields](#)

**Constructor**

```
ITransportProperties()
```

**Fields**

Name	Type	Description
<a href="#">id</a>	String	Transport vehicle identifier.
<a href="#">name</a>	String	Name of the transport vehicle's route.
<a href="#">type</a>	String	Transport vehicle type identifier. Accepts one of the following string values: <ul style="list-style-type: none"><li>"bus" - Bus.</li><li>"trolleybus" - Trolley.</li><li>"tramway" - Tram.</li><li>"minibus" - Minibus.</li><li>"underground" - Subway.</li><li>"suburban" - Commuter train.</li></ul>

**Fields details****id**

```
{String} id
```

Transport vehicle identifier.

**name**

```
{String} name
```

Name of the transport vehicle's route.

**type**

```
{String} type
```

Transport vehicle type identifier. Accepts one of the following string values:

- "bus" - Bus.
- "trolleybus" - Trolley.

- "tramway" - Tram.
- "minibus" - Minibus.
- "underground" - Subway.
- "suburban" - Commuter train.

## IZoomControlLayout

Extends [ILayout](#).

Interface for the layout of the "Zoom slider" control.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
IZoomControlLayout()
```

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .

### Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">emptinesschange</a>	Change to the empty layout indicator. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">ILayout</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">parentelementchange</a>	<p>Change to the parent element. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the layout. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">zoomchange</a>	<p>Event that initiates changing the map zoom level. Change the map zoom level. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>newZoom - New zoom level value.</li> <li>oldZoom - Old zoom level value.</li> </ul>

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		<p>Destructor. Called when activity with the layout is finished.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">getData()</a>	Object	<p>Returns layout data object.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">getParentElement()</a>	HTMLElement	<p>Returns parent HTML element.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	<p>Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">isEmpty()</a>	Boolean	<p>Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">setData(data)</a>		<p>Sets layout data.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">setParentElement(parent)</a>		<p>Adds the layout to the DOM tree.</p> <p>Inherited from <a href="#">ILayout</a>.</p>

## Events details

### zoomchange

Event that initiates changing the map zoom level. Change the map zoom level. Instance of the Event class. Names of fields that are available via the [Event.get](#) method:

- newZoom - New zoom level value.
- oldZoom - Old zoom level value.

## layer

### layer.storage

Static object.

Instance of [util.Storage](#)

Storage for layers.

[Methods](#)

#### Methods

Name	Returns	Description
<a href="#">add(key, object)</a>	<a href="#">util.Storage</a>	Adds an object to storage.
<a href="#">get(key)</a>	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
<a href="#">remove(key)</a>	<a href="#">util.Storage</a>	Deletes the "key: value" pair from storage.

### layer.tile

#### layer.tile.CanvasTile

Extends [ICanvasTile](#).

Image canvas tile. Can draw the specified image via the drawImage method for the 2d context of the canvas element.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
layer.tile.CanvasTile(url[, options[, renderOptions]])
```

Creates an image canvas tile. Accessible in the storage for tile classes by the key "default#canvas".

#### Parameters:

Parameter	Default value	Description
<a href="#">url</a> *	—	Type: String URL of the image.
<a href="#">options</a>	—	Type: Object Options.

Parameter	Default value	Description
<a href="#">options.notFoundTile</a>	null	Type: String null  Option that specifies the URL for downloading an image if the tile image didn't load. If the value is null, a standard tile is displayed with a text message. For transparent tiles, the notFoundTile option is not applied, and nothing is shown in place of tiles that didn't load.
<a href="#">options.tileAnimationDuration</a>	—	Type: Number  Duration of animation of tile transparency. The default value depends on the browser.
<a href="#">renderOptions</a>	—	Type: Object  Rendering parameters.
<a href="#">renderOptions.tileNumber</a>	—	Type: Number[]
<a href="#">renderOptions.tileZoom</a>	—	Type: Number

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

## Events

Name	Description
<a href="#">ready</a>	Tile ready event.  Inherited from <a href="#">ITile</a> .

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destroys the tile.  Inherited from <a href="#">ITile</a> .
<a href="#">isReady()</a>	Boolean	Checks tile readiness.  Inherited from <a href="#">ITile</a> .
<a href="#">renderAt(context, canvasSize, bounds[, animate])</a>		Draws an image tile in the canvas object's 2d context.  Inherited from <a href="#">ICanvasTile</a> .

**layer.tile.DomTile**

Extends [IDomTile](#).

Image tile. Can draw the specified image via the CSS "background" property of the DOM element.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
layer.tile.DomTile(url[, options[, renderOptions]])
```

Creates an image DOM tile. Accessible in the storage for tile classes by the key "default#dom". Creates an image DOM tile.

**Parameters:**

Parameter	Default value	Description
<a href="#">url</a> *	—	Type: String  URL of the image.
<a href="#">options</a>	—	Type: Object  Options.
<a href="#">options.notFoundTile</a>	null	Type: String null  Option that specifies the URL for downloading an image if the tile image didn't load. If the value is null, a standard tile is displayed with a text message. For transparent tiles, the notFoundTile option is not applied, and nothing is shown in place of tiles that didn't load.
<a href="#">options.tileAnimationDuration</a>	—	Type: Number  Duration of animation of image transparency when drawing, in ms (only applies in browsers that support CSS Transition for the "opacity" property). The default value depends on the browser.
<a href="#">renderOptions</a>	—	Type: Object  Rendering parameters.
<a href="#">renderOptions.tileNumber</a>	—	Type: Number[]
<a href="#">renderOptions.tileZoom</a>	—	Type: Number

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

**Events**

Name	Description
<a href="#">ready</a>	Tile ready event. Inherited from <a href="#">ITile</a> .

**Methods**

Name	Returns	Description
<a href="#">destroy()</a>		Destroys the tile. Inherited from <a href="#">ITile</a> .
<a href="#">isReady()</a>	Boolean	Checks tile readiness. Inherited from <a href="#">ITile</a> .
<a href="#">renderAt(context, clientBounds, animate)</a>		Adds a tile to the parent HTML element. Inherited from <a href="#">IDomTile</a> .

**layer.tileContainer****layer.tileContainer.CanvasContainer**

Extends [IChildOnMap](#).

Container for tiles on canvas.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
layer.tileContainer.CanvasContainer(layer[, options])
```

Creates a container for tiles on canvas. Accessible in the storage for tile container classes by the key "default#canvas".

**Parameters:**

Parameter	Default value	Description
<a href="#">layer</a> *	—	Type: <a href="#">ILayer</a> Layer.
<a href="#">options</a>	—	Type: Object Container options.
<a href="#">options.notFoundTile</a>	null	Type: String null Option that specifies the URL for downloading an image if the tile image didn't load. If the value is null, a standard tile is displayed with a text message. For transparent tiles, the notFoundTile option is not applied, and nothing is shown in place of tiles that didn't load.

Parameter	Default value	Description
<a href="#">options.tileClass</a>	'default#canvas'	Type: <a href="#">ICanvasTile</a>  Class of tiles used by the container. Must implement the interface <a href="#">ICanvasTile</a> .
<a href="#">options.tileTransparent</a>	false	Type: Boolean  Flag showing whether container tiles are transparent.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

## Events

Name	Description
<a href="#">parentchange</a>	The parent object reference changed.  Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map.
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IChildOnMap</a> .
<a href="#">getTile(tileNumber, tileZoom, priority)</a>	<a href="#">ICanvasTile</a>	Factory function for creating tiles.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .

## Methods details

### getMap

```
{Map} getMap ()
```

**Returns** reference to the map.

## getTile

```
{ICanvasTile} getTile(tileNumber, tileZoom, priority)
```

Factory function for creating tiles.

**Returns** tile instance.

### Parameters:

Parameter	Default value	Description
<code>tileNumber</code> *	—	Type: Number[] Tile number.
<code>tileZoom</code> *	—	Type: Number Tile scale.
<code>priority</code> *	—	Type: Number Loading priority.

\* Mandatory parameter/option.

## layer.tileContainer.DomContainer

Extends [IChildOnMap](#).

Container for tiles of the IDomTile type.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
layer.tileContainer.DomContainer(layer[, options])
```

Creates a container for DOM tiles. Accessible in the storage for tile container classes by the key "default#dom".

### Parameters:

Parameter	Default value	Description
<code>layer</code> *	—	Type: <a href="#">ILayer</a> Layer.
<code>options</code>	—	Type: Object Container options.
<code>options.notFoundTile</code>	null	Type: String null Option that specifies the URL for downloading an image if the tile image didn't load. If the value is null, a standard tile is displayed with a text message. For transparent tiles, the notFoundTile option is not applied, and nothing is shown in place of tiles that didn't load.

Parameter	Default value	Description
<a href="#">options.tileClass</a>	'default#dom'	Type: <a href="#">IDomTile</a>  Class of tiles used by the container. Must implement the interface <a href="#">IDomTile</a> .
<a href="#">options.tileTransparent</a>	false	Type: Boolean  Flag showing whether container tiles are transparent.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

## Events

Name	Description
<a href="#">parentchange</a>	The parent object reference changed.  Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .
<a href="#">ready</a>	Ready event for all tiles.

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map.
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IChildOnMap</a> .
<a href="#">getTile(tileNumber, tileZoom, priority)</a>	<a href="#">IDomTile</a>	Factory function for creating tiles.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .

## Events details

### ready

Ready event for all tiles.

## Methods details

### getMap

```
{Map} getMap()
```

**Returns** reference to the map.

### getTile

```
{IDomTile} getTile(tileNumber, tileZoom, priority)
```

Factory function for creating tiles.

**Returns** tile instance.

#### Parameters:

Parameter	Default value	Description
<a href="#">tileNumber</a> *	—	Type: Number[] Tile number.
<a href="#">tileZoom</a> *	—	Type: Number Tile scale.
<a href="#">priority</a> *	—	Type: Number Loading priority.

\* Mandatory parameter/option.

## Layer

Extends [ILayer](#), [IParentOnMap](#), [IPositioningContext](#).

Tile layer. Allows to display a layer consisting of tiles on the map.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
Layer(tileUrlTemplate[, options])
```

#### Parameters:

Parameter	Default value	Description
<a href="#">tileUrlTemplate</a> *	—	<p>Type: <code>String Function</code></p> <p>String template for the tile URL, or a function that generates the tile URL. For the string template, the following substitutions are supported:</p> <ul style="list-style-type: none"> <li>• <code>%c</code> is replaced with <code>x=number[0] &amp; y=number[1] &amp; z=zoom level</code>.</li> <li>• <code>%x</code> is replaced with <code>number[0]</code>.</li> <li>• <code>%y</code> is replaced with <code>number[1]</code>.</li> <li>• <code>%z</code> is replaced with the zoom level.</li> <li>• <code>%l</code> is replaced with <code>lang=language</code>.</li> <li>• <code>%d</code> or <code>%d n</code> is replaced with a number from 1 to <code>n</code>, depending on the tile number; <code>n</code> is the number of domains. Used for distributing the load over multiple domains. For <code>n</code>, specify a factor of two (2, 4, 16, and so on). If the template has <code>%d</code>, then <code>n=4</code>.</li> </ul> <p>The template function receives three input parameters:</p> <ul style="list-style-type: none"> <li>• <code>tileNumber</code> - Array of two numbers, the tile numbers on <code>x</code> and <code>y</code>.</li> <li>• <code>tileZoom</code> - Zoom level.</li> <li>• Returns a URL string.</li> </ul>
<a href="#">options</a>	—	<p>Type: <code>Object</code></p> <p>Options.</p>
<a href="#">options.brightness</a>	0.5	<p>Type: <code>Number</code></p> <p>Layer brightness. Specified as a number from 0 to 1. 0 corresponds to black, and 1 to white.</p>
<a href="#">options.notFoundTile</a>	null	<p>Type: <code>String null</code></p> <p>Option that specifies the URL for downloading an image if the tile image didn't load. If the value is null, a standard tile is displayed with a text message. For transparent tiles, the <code>notFoundTile</code> option is not applied, and nothing is shown in place of tiles that didn't load.</p>
<a href="#">options.pane</a>	'ground'	<p>Type: <code>IPane string</code></p> <p>Pointer to the layer pane or key from <a href="#">map.pane.Manager</a>.</p>
<a href="#">options.projection</a>	—	<p>Type: <code>Object</code></p> <p>Layer projection.</p>

Parameter	Default value	Description
<a href="#">options.tileSize</a>	[256, 256]	Type: Number[] Size of tiles on the layer.
<a href="#">options.tileTransparent</a>	false	Type: Boolean Flag showing whether layer tiles are transparent.
<a href="#">options.zIndex</a>	constants.zIndex.layer	Type: Number Z-index of the layer in the layers container.

\* Mandatory parameter/option.

#### Example:

```
// Adds an OSM layer to the map.
map.layers.add(new ymaps.Layer('http://tile.openstreetmap.org/%z/%x/%y.png', {
  projection: ymaps.projection.sphericalMercator
}));
map.copyrights.add('&copy; OpenStreetMap contributors, CC-BY-SA');
```

#### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

#### Events

Name	Description
<a href="#">brightnesschange</a>	Layer brightness change event. Inherited from <a href="#">ILayer</a> .
<a href="#">copyrightschange</a>	Event for changes to available copyright information. Inherited from <a href="#">ILayer</a> .
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> Inherited from <a href="#">IParentOnMap</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .

Name	Description
<a href="#">tileloadchange</a>	<p>Tile upload status change event. Data fields:</p> <ul style="list-style-type: none"> <li>readyTileNumber - Number of ready tiles. A tile is considered ready when it is downloaded and rendered. Type: Number.</li> <li>totalTileNumber - Total number of visible tiles. Type: Number.</li> </ul> <p>Inherited from <a href="#">ILayer</a>.</p>
<a href="#">zoomrangechange</a>	<p>Event for changes to available information about the zoom level range.</p> <p>Inherited from <a href="#">ILayer</a>.</p>

## Methods

Name	Returns	Description
<a href="#">clientPixelsToNumber(clientPixelPoint, tileZoom)</a>	Number[]	Returns the number of the tile that the specified point falls on for the specified tile zoom level.
<a href="#">fromClientPixels(clientPixelPoint)</a>	Number[]	<p>Converts client pixel coordinates to global coordinates.</p> <p>Inherited from <a href="#">IPositioningContext</a>.</p>
<a href="#">getBrightness()</a>	Number	<p>Optional method.</p> <p>Inherited from <a href="#">ILayer</a>.</p>
<a href="#">getCopyrights(coords, zoom)</a>	<a href="#">vow.Promise</a>	<p>Optional method. Requests information about copyrights at the specified point with the specified zoom.</p> <p>Inherited from <a href="#">ILayer</a>.</p>
<a href="#">getMap()</a>	<a href="#">Map</a>	<p>Returns reference to the map.</p> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">getPane()</a>	<a href="#">IPane</a>	Returns the container that the layer is located in.
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	<p>Returns link to the parent object, or null if the parent element was not set.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>
<a href="#">getTileSize(zoom)</a>	Number[]	Returns the horizontal and vertical tile dimensions for the specified zoom level.
<a href="#">getTileStatus()</a>	Object	Returns the total number of visible tiles and the number of ready tiles. A tile is considered ready when it is downloaded and rendered.

Name	Returns	Description
<a href="#">getTileUrl(tileNumber, tileZoom)</a>	String null	Returns the tile URL by its number and zoom level, or null if there is no data for the requested section.
<a href="#">getTileUrlTemplate()</a>	String Function	Returns string template for the tile URL, or a function that generates it.
<a href="#">getZoom()</a>	Number	Returns the current zoom level at which the positioning context works.  Inherited from <a href="#">IPositioningContext</a> .
<a href="#">getZoomRange(point)</a>	<a href="#">vow.Promise</a>	Optional method. Checks the available range of zoom levels at the specified point. If there is data, the returned promise object will be resolved and will pass as a result an array of two numbers - the minimum and maximum zoom level available at the point. If there is no data, the promise is rejected with an error.  Inherited from <a href="#">ILayer</a> .
<a href="#">numberToClientBounds(tileNumber, tileZoom)</a>	Number[][]	Converts the tile number and zoom level to the area occupied by the tile in client coordinates of the parent container.
<a href="#">restrict(number, tileZoom)</a>	Integer[] null	Applies restrictions to the visible area for tiles (including map cycling on the x and y axes).
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .
<a href="#">setTileUrlTemplate(tileUrlTemplate)</a>		
<a href="#">toClientPixels(globalPixelPoint)</a>	Number[]	Converts global pixel coordinates to client coordinates.  Inherited from <a href="#">IPositioningContext</a> .
<a href="#">update()</a>		Deletes the old tiles and requests new ones.

## Methods details

### clientPixelsToNumber

```
{Number[]} clientPixelsToNumber(clientPixelPoint, tileZoom)
```

**Returns** the number of the tile that the specified point falls on for the specified tile zoom level.

#### Parameters:

Parameter	Default value	Description
<code>clientPixelPoint</code> *	—	Type: Number A point in client pixel coordinates.
<code>tileZoom</code> *	—	Type: Number Tile zoom level.

\* Mandatory parameter/option.

### getPane

```
{IPane} getPane()
```

**Returns** the container that the layer is located in.

### getTileSize

```
{Number[]} getTileSize(zoom)
```

**Returns** the horizontal and vertical tile dimensions for the specified zoom level.

#### Parameters:

Parameter	Default value	Description
<code>zoom</code> *	—	Type: Number Zoom value.

\* Mandatory parameter/option.

#### Example:

```
// Show tiles for a larger zoom level,
// stretched to twice their size up to 512x512 pixels.
// For example, to reduce traffic.
var layer = new ymaps.Layer('', {
  projection: ymaps.projection.sphericalMercator
});
layer.getTileUrl = function (tileNumber, zoom) {
  return [
    'http://tile.openstreetmap.org',
    Math.max(zoom - 1, 0), tileNumber[0], tileNumber[1]
  ].join('/') + '.png';
}
layer.getTileSize = function (zoom) {
  if (zoom == 0) {
    return [256, 256];
  }
  return [512, 512];
}
map.copyrights.add('&copy; OpenStreetMap contributors, CC-BY-SA');
```

## getTileStatus

```
{Object} getTileStatus()
```

Returns the total number of visible tiles and the number of ready tiles. A tile is considered ready when it is downloaded and rendered.

**Returns** object with following fields:

- `readyTileNumber` - Number of ready tiles. Type: Number.
- `totalTileNumber` - Total number of tiles. Type: Number.

## getTileUrl

```
{String|null} getTileUrl(tileNumber, tileZoom)
```

**Returns** the tile URL by its number and zoom level, or null if there is no data for the requested section.

**Parameters:**

Parameter	Default value	Description
<code>tileNumber</code> *	—	Type:
<code>tileZoom</code> *	—	Type:

\* Mandatory parameter/option.

**Example:**

```
// Defines the function for generating the tile URL.
var layer = new ymaps.Layer('');
layer.getTileUrl = function (tileNumber, zoom) {
  return [
    'http://tile.openstreetmap.org',
    zoom, tileNumber[0], tileNumber[1]
  ].join('/') + '.png';
}
```

## getTileUrlTemplate

```
{String|Function} getTileUrlTemplate()
```

**Returns** string template for the tile URL, or a function that generates it.

## numberToClientBounds

```
{Number[][]} numberToClientBounds(tileNumber, tileZoom)
```

Converts the tile number and zoom level to the area occupied by the tile in client coordinates of the parent container.

**Returns** the area in client pixel coordinates.

**Parameters:**

Parameter	Default value	Description
<code>tileNumber</code> *	—	Type: Integer[] Tile number.
<code>tileZoom</code> *	—	Type: Integer Tile zoom level.

\* Mandatory parameter/option.

### restrict

```
{Integer[]|null} restrict(number, tileZoom)
```

Applies restrictions to the visible area for tiles (including map cycling on the x and y axes).

**Returns** the new tile number calculated with restrictions, or null if the tile is not in the visible area.

#### Parameters:

Parameter	Default value	Description
<code>number</code> *	—	Type: Integer[] Tile number.
<code>tileZoom</code> *	—	Type: Integer Tile zoom level.

\* Mandatory parameter/option.

### setTileUrlTemplate

```
{ } setTileUrlTemplate(tileUrlTemplate)
```

#### Parameters:

Parameter	Default value	Description
<code>tileUrlTemplate</code> *	—	Type: String Function String template for the tile URL, or a function that generates it.

\* Mandatory parameter/option.

### update

```
{ } update()
```

Deletes the old tiles and requests new ones.

#### Parameters:

Parameter	Default value	Description
<code>updateBounds</code> *	—	Type:

\* Mandatory parameter/option.

## LayerCollection

Extends [ILayer](#), [IMapObjectCollection](#).

Collection of layers.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
LayerCollection([options])
```

Collection of layers.

**Parameters:**

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object Layer options.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

**Events**

Name	Description
<a href="#">add</a>	A child object was added. Inherited from <a href="#">ICollection</a> .
<a href="#">brightnesschange</a>	Layer brightness change event. Inherited from <a href="#">ILayer</a> .
<a href="#">copyrightschange</a>	Event for changes to available copyright information. Inherited from <a href="#">ILayer</a> .
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> Inherited from <a href="#">IParentOnMap</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .
<a href="#">remove</a>	A child object was deleted. Inherited from <a href="#">ICollection</a> .

Name	Description
<a href="#">tileloadchange</a>	<p>Tile upload status change event. Data fields:</p> <ul style="list-style-type: none"> <li>• <code>readyTileNumber</code> - Number of ready tiles. A tile is considered ready when it is downloaded and rendered. Type: Number.</li> <li>• <code>totalTileNumber</code> - Total number of visible tiles. Type: Number.</li> </ul> <p>Inherited from <a href="#">ILayer</a>.</p>
<a href="#">zoomrangechange</a>	<p>Event for changes to available information about the zoom level range.</p> <p>Inherited from <a href="#">ILayer</a>.</p>

## Methods

Name	Returns	Description
<a href="#">add(child)</a>	<a href="#">LayerCollection</a>	Adds a child object to the collection.
<a href="#">each(callback[, context])</a>		Iterates through all the items in the collection and calls a handler function for each of them.
<a href="#">getBrightness()</a>	Number	Returns layer brightness as a number from 0 to 1.
<a href="#">getCopyrights([coords[, zoom]])</a>	<a href="#">vow.Promise</a>	Requests information about copyrights at the specified point with the specified zoom. If the point and zoom are omitted, it uses the map center and zoom.
<a href="#">getIterator()</a>	<a href="#">Iterator</a>	Returns iterator for the collection.  Inherited from <a href="#">ICollection</a> .
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map.  Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IChildOnMap</a> .

Name	Returns	Description
<a href="#">getZoomRange([coords])</a>	<a href="#">vow.Promise</a>	Checks the available range of zoom levels at the specified point. If there is data, the returned promise object will be resolved and will pass as a result an array of two numbers - the minimum and maximum zoom level available at the point. If there is no data, the promise is rejected with an error. If the collection does not have a single descendant providing information about the zoom level range, the promise will be rejected with the "noProvider" message.
<a href="#">remove(child)</a>	<a href="#">LayerCollection</a>	Removes a child object from the collection.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .

## Methods details

### add

```
{LayerCollection} add(child)
```

Adds a child object to the collection.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<a href="#">child</a> *	—	Type: <a href="#">ILayer</a>  String  Layer to add (key from <a href="#">layer.storage</a> or an instance of the <a href="#">ILayer</a> class).

\* Mandatory parameter/option.

#### Example:

```
// Let's say we want to add several layers to our collection.
var layerCollection = new ymaps.LayerCollection();
var customLayer = new ymaps.Layer('http://tile.openstreetmap.org/%z/%x/%y.png', {
  projection: ymaps.projection.sphericalMercator
});
// We can use a key from layer.storage to set the layer.
var satelliteLayer = 'yandex#satellite';
// Adding layers to our collection.
layerCollection
  .add(customLayer)
  .add(satelliteLayer);
```

## each

```
{ } each(callback[, context])
```

Iterates through all the items in the collection and calls a handler function for each of them.

### Parameters:

Parameter	Default value	Description
<code>callback</code> *	—	Type: Function Handler function.
<code>context</code>	—	Type: Object Context for the function.

\* Mandatory parameter/option.

## getBrightness

```
{Number} getBrightness()
```

**Returns** layer brightness as a number from 0 to 1.

## getCopyrights

```
{vow.Promise} getCopyrights([coords[, zoom]])
```

Requests information about copyrights at the specified point with the specified zoom. If the point and zoom are omitted, it uses the map center and zoom.

**Returns** a Promise object that will be resolved and will pass as a result an array of strings or DOM elements with information about copyrights.

### Parameters:

Parameter	Default value	Description
<code>coords</code>	—	Type: Number[] The point on the map that copyright information is being requested for.
<code>zoom</code>	—	Type: Number The zoom level that copyright information is being requested for.

### Example:

```
// Let's say we have a service that can return copyrights
// using the coordinates and zoom level
myLayer.getCopyrights = function (coords, zoom) {
  var deferred = ymaps.vow.defer();
  $.ajax('url/to/copyrights/provider?ll=' +
    (coords || map.getCenter()).join(',') + '&z=' +
    (zoom || map.getZoom()),
    function (res) {
      deferred.resolve(res || []);
    });
  return deferred.promise();
};
```

## getZoomRange

```
{vow.Promise} getZoomRange([coords])
```

Checks the available range of zoom levels at the specified point. If there is data, the returned promise object will be resolved and will pass as a result an array of two numbers - the minimum and maximum zoom level available at the point. If there is no data, the promise is rejected with an error. If the collection does not have a single descendant providing information about the zoom level range, the promise will be rejected with the "noProvider" message.

**Returns** Promise object.

### Parameters:

Parameter	Default value	Description
<code>coords</code>	—	Type: <code>Number[]</code>  Coordinates of a point. If omitted, the current map center is used.

### Example:

```
// Assuming that our layer was drawn for the 2-15 zoom level for the entire earth
myLayer.getZoomRange = function () {
  return ymaps.vow.resolve([2, 15]);
}
```

## remove

```
{LayerCollection} remove(child)
```

Removes a child object from the collection.

**Returns** self-reference.

### Parameters:

Parameter	Default value	Description
<code>child</code> *	—	Type: <code>ILayer String</code>  Layer to delete (key string from <a href="#">layer.storage</a> or an instance of the <a href="#">ILayer</a> class).

\* Mandatory parameter/option.

## layout

### layout.Image

Extends [ILayout](#).

Class for creating layouts containing a picture.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
layout.Image(data)
```

Creates an instance of the picture layout.

#### Parameters:

Parameter	Default value	Description
<code>data *</code>	—	Type: <a href="#">ILayout</a> Layout data.
<code>data.options</code>	—	Type: <a href="#">ILayout</a> Layout options.
<code>data.options.imageClipRect</code>	—	Type: <code>Number[][]</code> Coordinates of the display area of the original image, in pixels.
<code>data.options.imageHref</code>	—	Type: <code>String</code> URL of the image file.
<code>data.options.imageOffset</code>	—	Type: <code>Number[]</code> Offset of the image relative to the anchor point.
<code>data.options.imageSize</code>	—	Type: <code>Number[]</code> Dimensions of the image layer.
<code>data.options.shape</code>	—	Type: <a href="#">IShape</a>   <code>Object</code>   <code>null</code> The hotspot shape. Can be set as an instance of a class that implements the <a href="#">IShape</a> interface or a JSON description of the pixel geometry of the icon. If not set, the rectangular shape based on the size and offset of the icon will be calculated automatically. The coordinates of the figure geometry are counted from the anchor point.

\* Mandatory parameter/option.

#### Example:

```
// Creating a round placemark with a 20-pixel radius.
var placemark = new ymaps.Placemark([59.936952, 30.343334], null, {
  iconLayout: 'default#image',
  iconImageHref: './images/roundImage.png',
  iconImageSize: [40, 40],
  iconImageOffset: [-20, -20],
  // Defining a hotspot on top of the image.
  iconShape: {
    type: 'Circle',
    coordinates: [0, 0],
    radius: 20
  }
});
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .

**Events**

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">emptinesschange</a>	Change to the empty layout indicator. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">ILayout</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseleave</a>	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mousemove</a>	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseup</a>	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">multitouchend</a>	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface. Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">parentelementchange</a>	<p>Change to the parent element. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the layout. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		<p>Destructor. Called when activity with the layout is finished.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">getData()</a>	Object	<p>Returns layout data object.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">getParentElement()</a>	HTMLElement	<p>Returns parent HTML element.</p> <p>Inherited from <a href="#">ILayout</a>.</p>

Name	Returns	Description
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element.  Inherited from <a href="#">ILayout</a> .
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.  Inherited from <a href="#">ILayout</a> .
<a href="#">setData(data)</a>		Sets layout data.  Inherited from <a href="#">ILayout</a> .
<a href="#">setParentElement(parent)</a>		Adds the layout to the DOM tree.  Inherited from <a href="#">ILayout</a> .

## layout.ImageWithContent

Extends [layout.Image](#).

Class for creating layouts consisting of an image and content.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
layout.ImageWithContent (data)
```

Creates an instance of the layout image with content.

### Parameters:

Parameter	Default value	Description
<a href="#">data</a> *	—	Type: <a href="#">ILayout</a>  Layout data.
<a href="#">data.options</a>	—	Type: <a href="#">ILayout</a>  Layout options.
<a href="#">data.options.contentLayout</a>	—	Type: <a href="#">Function</a>   <a href="#">String</a>  Content layout. (Type: constructor for an object with the <a href="#">ILayout</a> interface, or its key in the storage).

Parameter	Default value	Description
<a href="#">data.options.contentOffset</a>	—	Type: Number[]  Offset of the layer with content relative to the layer with the image.
<a href="#">data.options.contentSize</a>	—	Type: Number[]  Dimensions of the content layer.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IDomEventEmitter</a> .

## Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">emptinesschange</a>	Change to the empty layout indicator. Instance of the <a href="#">Event</a> class.  Inherited from <a href="#">ILayout</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseleave</a>	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">parentelementchange</a>	<p>Change to the parent element. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the layout. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">destroy()</a>		Destructor. Called when activity with the layout is finished.  Inherited from <a href="#">ILayout</a> .
<a href="#">getData()</a>	Object	Returns layout data object.  Inherited from <a href="#">ILayout</a> .
<a href="#">getParentElement()</a>	HTMLElement	Returns parent HTML element.  Inherited from <a href="#">ILayout</a> .
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element.  Inherited from <a href="#">ILayout</a> .
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.  Inherited from <a href="#">ILayout</a> .
<a href="#">setData(data)</a>		Sets layout data.  Inherited from <a href="#">ILayout</a> .
<a href="#">setParentElement(parent)</a>		Adds the layout to the DOM tree.  Inherited from <a href="#">ILayout</a> .

**layout.PieChart**

Extends [layout.templateBased.Base](#).

Pie chart layout. Available in the layout storage by the key 'default#pieChart'. The layout can be used as a tool for visualizing any data, or in combination with other visual API components such as placemarks, the clusterer, and the objects manager.

**Note:** Because diagrams are drawn using SVG technology, this layout doesn't work in browsers that don't support SVG, including IE8.

**See** [Placemark Clusterer ObjectManager RemoteObjectManager LoadingObjectManager](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
layout.PieChart(data)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">data *</a>	—	Type: Object  Layout data.
<a href="#">data.option.pieChartRadius</a>	$25 + 2 * \text{Math.log}(\text{sum})$	Type: Number Function  Radius of the diagram, in pixels. Set as a number, or as a function that accepts the layout's "properties" and "options" parameters and returns the diagram radius as a number. By default, the radius is defined as $25 + 2 * \text{Math.log}(\text{sum})$ pixels, where sum is the total weight of the sectors.
<a href="#">data.options</a>	—	Type: <a href="#">IOptionManager</a>  Options for rendering the layout.
<a href="#">data.options.pieChartCaptionMaxWidth</a>	200	Type: Number  Maximum size of the label (iconCaption), in pixels.
<a href="#">data.options.pieChartCoreFillStyle</a>	white	Type: String  Fill style for the core. Set as a string that encodes the fill color.
<a href="#">data.options.pieChartCoreRadius</a>	<a href="#">pieChartRadius</a> - 15	Type: Number Function  The radius of the central part of the layout, where the content is displayed. Set in the same way as <a href="#">pieChartRadius</a> — as a number or a function. By default, it takes a value that is 15 pixels less than <a href="#">pieChartRadius</a> .
<a href="#">data.options.pieChartStrokeStyle</a>	white	Type: String  The style for lines between sectors and the outline of the diagram. Set as a string that encodes the line color.
<a href="#">data.options.pieChartStrokeWidth</a>	2	Type: Number  Width of the sector dividing lines and diagram outline, in pixels. Set as an integer.
<a href="#">data.properties *</a>	—	Type: <a href="#">IDataManager</a>  Object  Properties of the geo object that is displayed using the layout.
<a href="#">data.properties.data *</a>	—	Type: Object[] Function  Statistical data to base the layout on. It should be an array of JSON objects with the "weight" and "color" fields or a function that returns this array.

Parameter	Default value	Description
<a href="#">data.properties.geoObjects</a>	—	Type: <a href="#">IGeoObject[]</a>  An array of geo objects in the cluster that needs to be displayed. Used if "properties.data" is missing. In this case, the layout traverses all geo objects, determines the value of the "iconColor" option for each object, and generates a diagram based on this data.
<a href="#">data.properties.iconCaption</a>	""	Type: String  Caption for the diagram.
<a href="#">data.properties.iconContent</a>	The sum of all the sectors	Type: String  The value that is written in the center of the diagram. If omitted, the sum of all the sectors is shown.

\* Mandatory parameter/option.

#### Example:

```
var geoObject = new ymaps.Placemark([55.25, 37.43], {
  // Data for generating a diagram.
  data: [
    { weight: 5, color: '#224080' },
    { weight: 3, color: '#408022' },
    { weight: 2, color: '#802240' }
  ]
}, {
  iconLayout: 'default#pieChart',
  // You can also use the "icon" prefix to redefine layout options.
  iconPieChartCoreRadius: 15
});

myMap.geoObjects.add(geoObject);
```

#### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IDomEventEmitter</a> .

#### Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">emptinesschange</a>	<p>Change to the empty layout indicator. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">parentelementchange</a>	<p>Change to the parent element. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the layout. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">build()</a>		<p>Builds a layout instance based on the template and adds it to the parent HTML element.</p> <p>Inherited from <a href="#">layout.templateBased.Base</a>.</p>
<a href="#">clear()</a>		<p>Removes layout content from DOM.</p> <p>Inherited from <a href="#">layout.templateBased.Base</a>.</p>
<a href="#">destroy()</a>		<p>Destructor. Called when activity with the layout is finished.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">getData()</a>	Object	<p>Returns layout data object.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">getParentElement()</a>	HTMLElement	<p>Returns parent HTML element.</p> <p>Inherited from <a href="#">ILayout</a>.</p>

Name	Returns	Description
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. By default, it tries to construct a shape via the "shape" option - <code>this.getData().options.get('shape')</code> . In the option, it can be set as an instance of a class implementing the <a href="#">IShape</a> interface, or as a JSON description of the shape's pixel geometry. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element.  Inherited from <a href="#">layout.templateBased.Base</a> .
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.  Inherited from <a href="#">ILayout</a> .
<a href="#">onSublayoutSizeChange(sublayoutInfo, nodeSizeByContent)</a>		Automatically called when resizing a nested layout, added with the <code>'observeSize'</code> option. Used to override specific classes of layouts to respond to the resizing of the content.  Inherited from <a href="#">layout.templateBased.Base</a> .
<a href="#">rebuild()</a>		Re-sets the layout.  Inherited from <a href="#">layout.templateBased.Base</a> .
<a href="#">setData(data)</a>		Sets layout data.  Inherited from <a href="#">ILayout</a> .
<a href="#">setParentElement(parent)</a>		Adds the layout to the DOM tree.  Inherited from <a href="#">ILayout</a> .

## layout.storage

Static object.

Instance of [util.AsyncStorage](#)

Storage for layout classes.

[Methods](#)

**Methods**

Name	Returns	Description
<a href="#">add(key, object)</a>	<a href="#">util.Storage</a>	Adds an object to storage.
<a href="#">define(key[, depends, resolveCallback[, context]])</a>	<a href="#">util.AsyncStorage</a>	Defines an asynchronous value in storage.
<a href="#">get(key)</a>	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
<a href="#">isDefined(key)</a>	Boolean	Checking if the key can be accessed in the storage.
<a href="#">remove(key)</a>	<a href="#">util.Storage</a>	Deletes the "key: value" pair from storage.
<a href="#">require(keys[, successCallback[, errorCallback[, context]])</a>	<a href="#">vow.Promise</a>	Async request to get values from the storage.

**layout.templateBased****layout.templateBased.Base**

Extends [ILayout](#).

Basic layout class based on templates. This class is used by the layout factory as a base for creating user layouts.

See [templateLayoutFactory](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
layout.templateBased.Base(data)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">data</a> *	—	Type: Object  Set of different types of data that are used for building a layout.

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IDomEventEmitter</a> .

## Events

Name	Description
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">emptinesschange</a>	<p>Change to the empty layout indicator. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">parentelementchange</a>	<p>Change to the parent element. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the layout. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">build()</a>		Builds a layout instance based on the template and adds it to the parent HTML element.
<a href="#">clear()</a>		Removes layout content from DOM.
<a href="#">destroy()</a>		<p>Destructor. Called when activity with the layout is finished.</p> <p>Inherited from <a href="#">ILayout</a>.</p>
<a href="#">getData()</a>	Object	<p>Returns layout data object.</p> <p>Inherited from <a href="#">ILayout</a>.</p>

Name	Returns	Description
<a href="#">getParentElement()</a>	HTMLElement	Returns parent HTML element.  Inherited from <a href="#">ILayout</a> .
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	Returns a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. By default, it tries to construct a shape via the "shape" option - <code>this.getData().options.get('shape')</code> . In the option, it can be set as an instance of a class implementing the <a href="#">IShape</a> interface, or as a JSON description of the shape's pixel geometry. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element.
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.  Inherited from <a href="#">ILayout</a> .
<a href="#">onSublayoutSizeChange(sublayoutInfo, nodeSizeByContent)</a>		Automatically called when resizing a nested layout, added with the <code>`observeSize`</code> option. Used to override specific classes of layouts to respond to the resizing of the content.
<a href="#">rebuild()</a>		Re-sets the layout.
<a href="#">setData(data)</a>		Sets layout data.  Inherited from <a href="#">ILayout</a> .
<a href="#">setParentElement(parent)</a>		Adds the layout to the DOM tree.  Inherited from <a href="#">ILayout</a> .

## Methods details

### build

```
{ } build()
```

Builds a layout instance based on the template and adds it to the parent HTML element.

### clear

```
{ } clear()
```

Removes layout content from DOM.

## getShape

```
{IShape|null} getShape()
```

**Returns** a shape that defines the area spanning the layout, or null if it is not possible to plot this shape. By default, it tries to construct a shape via the "shape" option - `this.getData().options.get('shape')`. In the option, it can be set as an instance of a class implementing the `IShape` interface, or as a JSON description of the shape's pixel geometry. Coordinates of the shape's geometry should be calculated from the anchor point of the parent layout element.

### Example:

```
// Creating a placemark and setting a circular interactive area for it.
var MyLayoutClass = ymaps.templateLayoutFactory.createClass('<div class="imageIcon">{{name}}</div>');
var myPlacemark = new ymaps.Placemark([22, 34], {name: 'Cafe Mayak'}, {
  iconLayout: MyLayoutClass,
  iconShape: {type: 'Circle', coordinates: [0, 0], radius: 20}
});
```

## onSublayoutSizeChange

```
{ } onSublayoutSizeChange(sublayoutInfo, nodeSizeByContent)
```

Automatically called when resizing a nested layout, added with the `'observeSize'` option. Used to override specific classes of layouts to respond to the resizing of the content.

### Parameters:

Parameter	Default value	Description
<code>sublayoutInfo</code> *	—	Type: Object  Information about the nested layout.
<code>nodeSizeByContent</code> *	—	Type: Object  The new size of the element considering the size of its content. Available fields: <code>'width'</code> , <code>'height'</code> , <code>'scrollX'</code> , <code>'scrollY'</code> .

\* Mandatory parameter/option.

## rebuild

```
{ } rebuild()
```

Re-sets the layout.

## LoadingObjectManager

Extends [ICustomizable](#), [IEventEmitter](#), [IGeoObject](#), [IParentOnMap](#).

The object manager that optimizes downloading of objects from the server. Allows optimally downloading, displaying, clustering and managing visibility for objects. The manager sends the data request to the specified url in the JSONP format. This format corresponds to the format of objects added to `ObjectManager`, [ObjectManager.add](#). Note that objects drawn on the map via this manager can't have editing and dragging modes enabled.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

## Constructor

```
LoadingObjectManager(urlTemplate[, options])
```

### Parameters:

Parameter	Default value	Description
<a href="#">urlTemplate</a> *	—	<p>Type: String</p> <p>URL data template. Supports special constructions similar to <a href="#">Layer</a>. Substitutions are also supported:</p> <ul style="list-style-type: none"> <li>• %b is replaced by an array of geographic coordinates that describes the rectangular region for which you want to load data.</li> <li>• %t is replaced by an array of tile numbers that describes the rectangular region to load data for.</li> </ul>
<a href="#">options</a>	—	<p>Type: Object</p> <p>Options.</p> <ul style="list-style-type: none"> <li>• You can set all the options specified in the <a href="#">Clusterer</a> description, except for hasBalloon and hasHint options.</li> <li>• Cluster options are set with the "cluster" prefix. The list of options is specified in the description of <a href="#">ClusterPlacemark</a>;</li> <li>• Options for singular objects should be specified with the geoObject prefix. The list of options is specified in <a href="#">GeoObject</a>. Note that the manager ignores the 'visible' option.</li> </ul>
<a href="#">options.clusterize</a>	false	<p>Type: Boolean</p> <p>Flag indicating whether the objects should be clusterized. Note that clusterization only works for point objects at this time. When cluster mode is enabled, all non-point objects are ignored.</p>
<a href="#">options.loadTileSize</a>	256	<p>Type: Number</p> <p>Tile size for data loading.</p>
<a href="#">options.paddingParamName</a>	'callback'	<p>Type: Boolean</p> <p>Name of the GET parameter that contains the value of the JSONP callback.</p>

Parameter	Default value	Description
<a href="#">options.paddingTemplate</a>	null	<p>Type: String</p> <p>Template for a jsonp callback. Supports the same substitutions as uriTemplate. All characters other than letters and numbers will be replaced with '_'. If the parameter is omitted, the name of the jsonp callback will be generated automatically. Conversion examples for tileNumber=[3, 1], zoom=9:</p> <ul style="list-style-type: none"> <li>'myCallback=%x' =&gt; 'myCallback_3'</li> <li>'%c' =&gt; 'x_3_y_1_z_9'</li> <li>'callback2_%c' =&gt; 'callback2_x_3_y_1_z_9'</li> <li>'callback%test' =&gt; 'callback_test'</li> <li>'callback_%b' =&gt; 'callback_85_0841__180_0000_85_0841_180_0000'</li> </ul> <p>Note that if substitution options are not used in the value, this may lead to an error. All requests will go to the same callback function.</p>
<a href="#">options.splitRequests</a>	false	<p>Type: Boolean</p> <p>Divide requests for data into requests for individual tiles. By default, requests are made for data for a rectangular region that contains multiple tiles.</p>
<a href="#">options.syncOverlayInit</a>	false	<p>Type: Boolean</p> <p>A flag that allows creating overlays for objects synchronously. Note that when you create an overlay synchronously, you should ensure that the appropriate class, which implements the IOverlay interface, is loaded. By default, the overlays are created asynchronously, and the overlay class is loaded on demand.</p>
<a href="#">options.viewportMargin</a>	128	<p>Type: Number Number[]</p> <p>Offset for the area where the objects are shown. Use this option to expand the view area of objects relative to the visible area of the map.</p>

\* Mandatory parameter/option.

### Examples:

#### 1.

```
var objectManager = new ymaps.LoadingObjectManager('http://myServer.com/tile?bbox=%b', {
  // Enabling clustering.
  clusterize: true,
  // Cluster options are set with the "cluster" prefix.
  clusterHasBalloon: false,
  // Geo object options are set with the "geoObject" prefix.
  geoObjectOpenBalloonOnClick: false
});

// You can set options directly for child collections.
objectManager.clusters.options.set({
```

```

    preset: 'islands#grayClusterIcons',
    hintContentLayout: ymaps.templateLayoutFactory.createClass('Group of objects')
  });
  objectManager.objects.options.set('preset', 'islands#grayIcon');

```

## 2.

```

An example of LoadingObjectManager response
jsonp_callback({
  // The response contains error and data fields. If an error occurs, the "error" field
  // contains the error code or description.
  error: null,
  data: {
    type: 'FeatureCollection',
    features: [
      {
        type: 'Feature',
        geometry: {
          type: 'Point',
          coordinates: [55, 35]
        },
        id: 23,
        properties: {
          balloonContent: 'Placemark balloon content',
          iconContent: 'Placemark content'
        },
        options: {
          preset: 'islands#yellowIcon'
        }
      }
    ]
  }
});

```

## Fields

Name	Type	Description
<a href="#">clusters</a>	<a href="#">objectManager.ClusterCollection</a>	Collection of clusters generated by the manager.
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">objects</a>	<a href="#">objectManager.ObjectCollection</a>	Collection of objects added to the layer.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">IDataManager</a>	Geo object data. Inherited from <a href="#">IGeoObject</a> .
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

## Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li></ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>oldMap - Old map.</li><li>newMap - New map.</li></ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getBounds()</a>	Number[][] null	Calculates the boundaries in geo coordinates for an area that covers all the loaded objects in the manager.
<a href="#">getMap()</a>	Map	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getObjectState(id)</a>	Object	Getting information about the current state of an object added to the manager.
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">getPixelBounds()</a>	Number[][] null	Calculates the boundaries in global pixel coordinates for an area that covers all the loaded objects in the manager.
<a href="#">getTileUrl()</a>	String null	Returns URL of the tile with data.
<a href="#">getUrlTemplate()</a>	String	Returns URL data template.
<a href="#">reloadData()</a>		Method that deletes all previously loaded data and sends a request for new data.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">setUrlTemplate(uriTemplate)</a>		

## Fields details

### clusters

```
{objectManager.ClusterCollection} clusters
```

Collection of clusters generated by the manager.

#### Example:

```
objectManager.clusters.events.add('click', function (e) {  
  var objectId = e.get('objectId');  
  objectManager.clusters.balloon.open(objectId);  
});
```

### objects

```
{objectManager.ObjectCollection} objects
```

Collection of objects added to the layer.

#### Example:

```
objectManager.objects.events.add('click', function (e) {  
  var objectId = e.get('objectId');  
  objectManager.objects.balloon.open(objectId);  
});
```

## Methods details

### getBounds

```
{Number[][]|null} getBounds()
```

Calculates the boundaries in geo coordinates for an area that covers all the loaded objects in the manager.

**Returns** array of the area's coordinates, or null if the manager has not been added to the map.

### getObjectState

```
{Object} getObjectState(id)
```

Getting information about the current state of an object added to the manager.

**Returns** object with following fields:

- **found** - Attribute that indicates whether an object with the passed ID exists. Type: Boolean.
- **isShown** - Attribute that indicates whether the object is located in the visible area of the map. Type: Boolean.
- **cluster** - JSON description of the cluster the object was added to. Besides the required fields, it contains the `properties.geoObjects` field with an array of objects that are in the cluster. This field is returned only when clusterization is enabled.
- **isClustered** - Attribute indicating whether an object is in the cluster. This field is returned only when clusterization is enabled. Type: Boolean.
- **isFilteredOut** - Attribute indicating whether an object passed through filtration. If the filter is not set or the object passed through filtration, the value of the field is "false". Type: Boolean.

#### Parameters:

Parameter	Default value	Description
<code>id</code> *	—	Type: Object  ID of the object to get the state for.

\* Mandatory parameter/option.

#### Example:

```
// Opening the cluster balloon with the selected object.
// Getting data about the state of an object inside the cluster.
var objectState = objectManager.getObjectState(objects[1].id);
// Checking whether the object is located in the visible area of the map.
if (objectState.found && objectState.isShown) {
  // If the object is in a cluster, we open the cluster balloon with the appropriate object selected.
  if (objectState.isClustered) {
    objectManager.clusters.state.set('activeObject', objects[1]);
    objectManager.clusters.balloon.open(objectState.cluster.id);
  } else {
    // If the object isn't in a cluster, we open its own balloon.
    objectManager.objects.balloon.open(objects[1].id);
  }
}
```

#### getPixelBounds

```
{Number[][]|null} getPixelBounds()
```

Calculates the boundaries in global pixel coordinates for an area that covers all the loaded objects in the manager.

**Returns** array of the area's coordinates, or null if the manager has not been added to the map.

#### getTileUrl

```
{String|null} getTileUrl()
```

**Returns** URL of the tile with data.

#### Parameters:

Parameter	Default value	Description
<a href="#">parameters</a> *	—	Type:

\* Mandatory parameter/option.

#### Example:

```
var objectManager = new ymaps.LoadingObjectManager('http://myServer.com/tile?bbox=%b');
objectManager.getTileUrl = function (parameters) {
  var boundingBox = parameters.boundingBox.join('~');
  return this.getUrlTemplate().replace(/%b/g, boundingBox);
};
```

#### getUrlTemplate

```
{String} getUrlTemplate()
```

**Returns** URL data template.

#### reloadData

```
{} reloadData()
```

Method that deletes all previously loaded data and sends a request for new data.

#### setUrlTemplate

```
{} setUrlTemplate(urlTemplate)
```

#### Parameters:

Parameter	Default value	Description
<a href="#">uriTemplate</a> *	—	Type: String  URL data template.

\* Mandatory parameter/option.

## Map

Extends [IDomEventEmitter](#).

Class for creating and managing a map.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
Map(parentElement, state[, options])
```

Creates a map instance.

### Parameters:

Parameter	Default value	Description
<a href="#">parentElement</a> *	—	Type: Object String  Reference to an HTML element that contains the map, or the ID of this HTML element.
<a href="#">state</a> *	—	Type: Object  Map parameters.
<a href="#">state.behaviors</a>	['default']	Type: String[]  Enabled behaviors for the map. By default, the following are enabled: dragging the map and zooming the map by multitouch or double-click on touch screen devices; dragging the map with the mouse, double-click zooming and right-click area selection on all other devices. You can specify any keys supported by <a href="#">behavior.Manager</a> .
<a href="#">state.bounds</a>	—	Type: Number[][]  Geo coordinates of the viewport the map starts with. When initializing the map, you can set either a <a href="#">state.zoom-state.center</a> pair, or this viewport. If <a href="#">state.bounds</a> is set and the parameters <a href="#">state.zoom</a> or <a href="#">state.center</a> are also set, they will be ignored.
<a href="#">state.center</a>	—	Type: Number[]  Geo coordinates of the map center. They must be set in conjunction with <a href="#">state.zoom</a> .

Parameter	Default value	Description
<a href="#">state.controls</a>	['default']	Type: String[]  The map controls, added by default. You can specify any keys supported by <a href="#">control.Manager</a> . The list of available keys and default sets of controls are described in <a href="#">control.Manager.add</a> .
<a href="#">state.margin</a>	—	Type: Number Number[]  Offsets from the map edges. Passed to <a href="#">map.margin.Manager.setDefaultMargin</a> .
<a href="#">state.type</a>	'yandex#map'	Type: String MapType  Map type. Can be a key or an instance of the <a href="#">MapType</a> class. List of available keys: <ul style="list-style-type: none"> <li>• 'yandex#map' - "Roadmap" map type.</li> <li>• 'yandex#satellite' - "Satellite" map type.</li> <li>• 'yandex#hybrid' - "Hybrid" map type.</li> </ul>
<a href="#">state.zoom</a>	—	Type: Number  Map zoom level. It must be set in conjunction with <a href="#">state.center</a> .
<a href="#">options</a>	—	Type: Object  Map options. The map options can be used to make settings for the map itself, as well as for objects that are added to it: <ul style="list-style-type: none"> <li>• Options for <a href="#">map behaviors</a>.</li> <li>• Options for the <a href="#">map balloon</a> with the <code>balloon</code> prefix.</li> <li>• Options for the <a href="#">map hint</a> with the <code>hint</code> prefix.</li> <li>• Options for <a href="#">geo objects</a> with the <code>geoObject</code> prefix.</li> <li>• Options for <a href="#">layers</a> with the <code>layer</code> prefix.</li> <li>• Options for <a href="#">hotspot layers</a> with the <code>hotspotLayer</code> prefix.</li> </ul> Options that are interpreted directly by the map itself are listed below.

Parameter	Default value	Description
<a href="#">options.autoFitToViewport</a>	'ifNull'	<p>Type: String</p> <p>Automatic map container tracking. By default, the map is automatically redrawn when it is initialized from a hidden container, or if we programmatically changed its dimensions; otherwise, you must call <code>map.container.fitToViewport</code>. The following values are available:</p> <ul style="list-style-type: none"><li>'none' — Don't track changes to the container display when initialized from a hidden container or when its size is changed programmatically.</li><li>'ifNull' — As soon as the container gets a CSS "display" value other than "none", <code>map.container.fitToViewport</code> executes automatically in order to fit the map to it. After this, tracking will stop.</li><li>'always' — Always track changes to the display state of the map container.</li></ul>
<a href="#">options.avoidFractionalZoom</a>	true	<p>Type: Boolean</p> <p>If true, the map does not stop on fractional values of the zoom level; if false, it does.</p> <p><b>Note:</b> By default, this option is true for desktop browsers and false for mobile browsers.</p>
<a href="#">options.exitFullscreenByEsc</a>	true	<p>Type: Boolean</p> <p>Enables to exit full-screen mode using the ESC button. If true, the map exits full-screen mode when the ESC button is pressed; if false, it does not.</p>
<a href="#">options.fullscreenZIndex</a>	10000	<p>Type: Number</p> <p>The value of the z-index property of the map container in full-screen mode.</p>
<a href="#">options.mapAutoFocus</a>	true	<p>Type: Boolean</p> <p>If true, clicking on the map will move the focus out of the currently active DOM element; if false, it will keep the focus on the active element.</p>
<a href="#">options.maxAnimationZoomDifference</a>	5	<p>Type: Number</p> <p>The maximum difference between the current map zoom level and the new zoom level so that zooming will look smooth.</p>

Parameter	Default value	Description
<a href="#">options.maxZoom</a>	23	Type: Number  Maximum map zoom level.
<a href="#">options.minZoom</a>	0	Type: Number  Minimum map zoom level.
<a href="#">options.nativeFullscreen</a>	false	Type: Boolean  To use the native fullscreen "mode" if possible. Switching to the native full screen mode means that the browser asks the user if they want to go into "fullscreen mode" and if the user agrees, expands the map to full screen, hides browser controls and allows the user to exit the mode by pressing ESC key. The <code>fullscreenZIndex</code> and <code>exitFullscreenByEsc</code> options have no effect in the native "fullscreen mode".
<a href="#">options.projection</a>	<code>ymaps.projection.wgs84Mercator</code>	Type: <a href="#">IProjection</a>  Map projection.
<a href="#">options.restrictMapArea</a>	false	Type: Boolean Number[][]  Sets the map viewport so that the user cannot leave the viewport area. True — Set the area to match the current display area; false — disable this option. If you need to restrict an area with known coordinates, specify this rectangular area.
<a href="#">options.suppressMapOpenBlock</a>	false	Type: Boolean  Whether to hide the offer to open the current map in Yandex.Maps with all the available map information preserved as completely as possible. True - hide; false - don't hide. The link to Yandex.Maps is displayed in the lower-left corner of the map.
<a href="#">options.suppressObsoleteBrowserNotifier</a>	false	Type: Boolean  Whether to hide the notification suggesting a browser upgrade that is shown in outdated browsers. True - hide; false - don't hide. This notification is displayed in the lower-left corner of the map.
<a href="#">options.yandexMapAutoSwitch</a>	true	Type: Boolean  true - Enable automatically switching to the Public Map in those places where the map is not detailed enough; false - disable. This option only works for <code>lang=ru_RU</code> and <code>lang=uk_UA</code> .

Parameter	Default value	Description
<a href="#">options.yandexMapDisablePoiInteractivity</a>	false	Type: boolean  True - disable interactivity of POI (points of interest) in the map's base layer, false — enable.

\* Mandatory parameter/option.

### Examples:

#### 1.

```
// Initializing a map with a known center and zoom level
var myMap = new ymaps.Map('map', {
  center: [55.74954, 37.621587],
  zoom: 10
});
```

#### 2.

```
// Initializing a map by a known view area
// We assume that jQuery is enabled on the page
var $mapElement = $('#map');
var myMap = new ymaps.Map(
  $mapElement[0],
  ymaps.util.bounds.getCenterAndZoom(
    [[55.7, 37.6], [55.8, 37.7]],
    [$mapElement.width(), $mapElement.height()]
  )
);
```

#### 3.

```
// Initializing a map from geocoding results
var myMap;
ymaps.geocode('Moscow').then(function (res) {
  myMap = new ymaps.Map('map', {
    center: res.geoObjects.get(0).geometry.getCoordinates(),
    zoom : 10
  });
});
```

### Fields

Name	Type	Description
<a href="#">action</a>	<a href="#">map.action.Manager</a>	Map actions manager.
<a href="#">balloon</a>	<a href="#">map.Balloon</a>	Map balloon.
<a href="#">behaviors</a>	<a href="#">map.behavior.Manager</a>	Map behaviors manager. Enables and disables behaviors, and also provides access to their methods and properties.
<a href="#">container</a>	<a href="#">map.Container</a>	Map container.
<a href="#">controls</a>	<a href="#">control.Manager</a>	Map controls.
<a href="#">converter</a>	<a href="#">map.Converter</a>	Converts map pixel points from global to local and back.
<a href="#">copyrights</a>	<a href="#">map.Copyrights</a>	Manager for copyright information on the map.
<a href="#">cursors</a>	<a href="#">util.cursor.Manager</a>	Map cursors manager.
<a href="#">events</a>	<a href="#">event.Manager</a>	Map event manager. Supports subscriptions with priorities. Throws an event of the <a href="#">MapEvent</a> type.
<a href="#">geoObjects</a>	<a href="#">map.GeoObjects</a>	Manager for map geo objects.

Name	Type	Description
<a href="#">hint</a>	<a href="#">map.Hint</a>	Map hint.
<a href="#">layers</a>	<a href="#">map.layer.Manager</a>	Map layers manager.
<a href="#">margin</a>	<a href="#">map.margin.Manager</a>	Map margins manager.
<a href="#">options</a>	<a href="#">option.Manager</a>	Map options.
<a href="#">panes</a>	<a href="#">map.pane.Manager</a>	Manager for map object containers.
<a href="#">zoomRange</a>	<a href="#">map.ZoomRange</a>	Object that provides access to information about available zoom levels at a point.

## Events

Name	Description
<a href="#">actionbegin</a>	The start of a new smooth map movement. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>action</code> - Action that has started.</li> </ul>
<a href="#">actionbreak</a>	Event that occurs when an action step was prematurely stopped (for example, because another action or another step of the action was performed). Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>action</code> - An action.</li> </ul>
<a href="#">actionend</a>	The end of smooth map movement. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>action</code> - Action that has stopped.</li> </ul>
<a href="#">actiontick</a>	The start of a new step of smooth movement (for example, the user shifting the map, or a step of animating smooth zooming). Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>action</code> - The action being performed at this moment.</li> <li><code>tick</code> - Description of an action step in the form of an object with the fields "globalPixelCenter", "zoom", "duration" and "timingFunction".</li> </ul>
<a href="#">actiontickcomplete</a>	The end of performing a step of smooth movement. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>action</code> - The action being performed at this moment.</li> <li><code>tick</code> - Description of an action step in the form of an object with the fields "globalPixelCenter", "zoom", "duration" and "timingFunction".</li> </ul>
<a href="#">balloonclose</a>	Closing the balloon. Instance of the <a href="#">Event</a> class.
<a href="#">balloonopen</a>	Opening a balloon on a map. Instance of the <a href="#">Event</a> class.

Name	Description
<a href="#">boundschange</a>	<p>Event for changing the map viewport (as the result of changing the center or zoom level). Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>oldCenter</code> - The previous map center, in geocoordinates.</li><li>• <code>newCenter</code> - The new map center, in geocoordinates.</li><li>• <code>oldZoom</code> - The previous zoom level.</li><li>• <code>newZoom</code> - The new zoom level.</li><li>• <code>oldGlobalPixelCenter</code> - The previous map center, in global pixels.</li><li>• <code>newGlobalPixelCenter</code> - The new map center, in global pixels.</li><li>• <code>oldBounds</code> - Previous viewport.</li><li>• <code>newBounds</code> - New viewport.</li></ul>
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">destroy</a>	<p>The map was destroyed.</p>
<a href="#">hintclose</a>	<p>Closing the hint. Instance of the <a href="#">Event</a> class.</p>
<a href="#">hintopen</a>	<p>Opening a hint on a map. Instance of the <a href="#">Event</a> class.</p>
<a href="#">marginchange</a>	<p>Map margins changed. Instance of the <a href="#">Event</a> class.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">mouseup</a>	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">multitouchend</a>	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">multitouchmove</a>	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields: <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">multitouchstart</a>	Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields: <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">optionschange</a>	Map options changed.
<a href="#">sizechange</a>	Map size changed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>oldSize</code> - The previous size of the map.</li> <li><code>newSize</code> - The new size of the map.</li> </ul>
<a href="#">typechange</a>	The map type changed. Instance of the <a href="#">Event</a> class.
<a href="#">wheel</a>	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destroys the map.

Name	Returns	Description
<a href="#">getBounds([options])</a>	Number[][]	Returns a two-dimensional array of geocoordinates for the lower-left and upper-right corners of the map viewport.
<a href="#">getCenter([options])</a>	Number[]	Returns geographical coordinates of the current map center.
<a href="#">getGlobalPixelCenter([options])</a>	Number[]	Returns global pixel coordinates of the current map center.
<a href="#">getPanoramaManager()</a>	<a href="#">vow.Promise</a> .< <a href="#">panorama.Manager</a> >	Returns <a href="#">vow.Promise</a> , which will be resolved with The panorama manager for the map. If an error occurs, the promise object is rejected.
<a href="#">getType()</a>	String  <a href="#">MapType</a>	Returns the current map type.
<a href="#">getZoom()</a>	Number	Returns the current map zoom.
<a href="#">panTo(center[, options])</a>	<a href="#">vow.Promise</a>	Sets the map center. If an array of points is passed, the map will move from one point to the other.
<a href="#">setBounds(bounds[, options])</a>	<a href="#">vow.Promise</a>	Positions the map for displaying the region that was passed.
<a href="#">setCenter(center[, zoom[, options]])</a>	<a href="#">vow.Promise</a>	Sets the map center and zoom level. The center is set in geographical coordinates.
<a href="#">setGlobalPixelCenter(globalPixelCenter[, zoom[, options]])</a>	<a href="#">vow.Promise</a>	Sets the map center and zoom level. The center is set in global pixel coordinates.
<a href="#">setType(type[, options])</a>	<a href="#">vow.Promise</a>	Sets the map type.
<a href="#">setZoom(zoom[, options])</a>	<a href="#">vow.Promise</a>	Sets the map zoom level.

## Fields details

### action

```
{map.action.Manager} action
```

Map actions manager.

### Example:

```
var myAction = new ymaps.map.action.Single({
  center: [0, 0],
  zoom: 4,
  duration: 1000,
  timingFunction: "ease-in"
});
myMap.action.execute(myAction);
```

## balloon

```
{map.Balloon} balloon
```

Map balloon.

## behaviors

```
{map.behavior.Manager} behaviors
```

Map behaviors manager. Enables and disables behaviors, and also provides access to their methods and properties.

### Example:

```
// Enabling mouse wheel zooming  
myMap.behaviors.enable('scrollZoom');
```

## container

```
{map.Container} container
```

Map container.

### Example:

```
// Adjusting the map size to the new size of the container  
// (for example, if the page layout changed or the map was initialized  
// in the hidden state)  
map.container.fitToViewport();
```

## controls

```
{control.Manager} controls
```

Map controls.

### Example:

```
myMap.controls.add('zoomControl', {  
  float: 'none',  
  position: {  
    right: 40,  
    top: 5  
  }  
});
```

## converter

```
{map.Converter} converter
```

Converts map pixel points from global to local and back.

### Example:

```
// Converting the mouse coordinates to geographical coordinates  
var projection = map.options.get('projection');  
$('#map').bind('click', function (e) {  
  console.log(projection.fromGlobalPixels(  
    map.converter.pageToGlobal([e.pageX, e.pageY]), map.getZoom()  
  ));  
});
```

## copyrights

```
{map.Copyrights} copyrights
```

Manager for copyright information on the map.

**Example:**

```
// Adding author information
map.copyrights.add('&copy; Jimmy John');
```

**cursors**

```
{util.cursor.Manager} cursors
```

Map cursors manager.

**Example:**

```
// Adding the "Help" cursor to the map
var accessor = map.cursors.push('help');
// ...
// Removing the cursor
accessor.remove();
```

**events**

```
{event.Manager} events
```

Map event manager. Supports subscriptions with priorities. Throws an event of the [MapEvent](#) type.

**Examples:****1.**

```
// Adding a placemark on a click on the map.
map.events.add('click', function (e) {
  // To get the geographical coordinates of the point of click,
  // call .get('coords')
  var position = e.get('coords');
  map.geoObjects.add(new ymaps.Placemark(position));
});
```

**2.**

```
// Tracking the map's center and zoom during smooth movements.
map.events.add('actiontick', function () {
  var state = map.action.getCurrentState();
  console.log(state.zoom, state.globalPixelCenter);
});
```

**geoObjects**

```
{map.GeoObjects} geoObjects
```

Manager for map geo objects.

**hint**

```
{map.Hint} hint
```

Map hint.

**layers**

```
{map.layer.Manager} layers
```

Map layers manager.

See [Layer](#)

**Example:**

```
// Adding a layer of custom tiles to the map
map.layers.add(new ymaps.Layer('http://some.server/tiles?&map=c'));
```

**margin**

```
{map.margin.Manager} margin
```

Map margins manager.

**options**

```
{option.Manager} options
```

Map options.

**Example:**

```
// Forbidding all objects on the map to open balloons on mouse clicks.
map.options.set('openBalloonOnClick', false);
```

**panes**

```
{map.pane.Manager} panes
```

Manager for map object containers.

**Example:**

```
// Adding a custom element to the map controls container
$('<div><input type="button" value="Click!"/></div>')
  .css({ position: 'absolute', left: '5px', top: '50px'})
  .appendTo(map.panes.get('controls').getElement());
```

**zoomRange**

```
{map.ZoomRange} zoomRange
```

Object that provides access to information about available zoom levels at a point.

**Example:**

```
// Getting the maximum available map zoom level
// at the point [20, 30]
map.zoomRange.get([20, 30]).then(function (zoomRange) {
  alert(zoomRange[1]);
});
```

**Events details****actionbegin**

The start of a new smooth map movement. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- action - Action that has started.

**actionbreak**

Event that occurs when an action step was prematurely stopped (for example, because another action or another step of the action was performed). Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- action - An action.

### actionend

The end of smooth map movement. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- action - Action that has stopped.

### actiontick

The start of a new step of smooth movement (for example, the user shifting the map, or a step of animating smooth zooming). Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- action - The action being performed at this moment.
- tick - Description of an action step in the form of an object with the fields "globalPixelCenter", "zoom", "duration" and "timingFunction".

#### Example:

```
// Tracks all map movement, even user dragging
// and smooth zooming
map.events.add('actiontick', function (e) {
  var tick = e.get('tick');
  console.log('Now the map is moving to the point (' +
    map.options.get('projection').fromGlobalPixels(tick.globalPixelCenter, tick.zoom).join(',') +
    ') during ' + e.get('tick').duration + ' milliseconds');
});
```

### actiontickcomplete

The end of performing a step of smooth movement. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- action - The action being performed at this moment.
- tick - Description of an action step in the form of an object with the fields "globalPixelCenter", "zoom", "duration" and "timingFunction".

### balloonclose

Closing the balloon. Instance of the [Event](#) class.

### balloonopen

Opening a balloon on a map. Instance of the [Event](#) class.

### boundschange

Event for changing the map viewport (as the result of changing the center or zoom level). Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- oldCenter - The previous map center, in geocoordinates.
- newCenter - The new map center, in geocoordinates.
- oldZoom - The previous zoom level.
- newZoom - The new zoom level.
- oldGlobalPixelCenter - The previous map center, in global pixels.
- newGlobalPixelCenter - The new map center, in global pixels.
- oldBounds - Previous viewport.
- newBounds - New viewport.

#### Example:

```
// Tracking changes to the map zoom level
map.events.add('boundschange', function (event) {
  if (event.get('newZoom') !== event.get('oldZoom')) {
    alert('Zoom level changed');
  }
});
```

**destroy**

The map was destroyed.

**hintclose**

Closing the hint. Instance of the [Event](#) class.

**hintopen**

Opening a hint on a map. Instance of the [Event](#) class.

**marginchange**

Map margins changed. Instance of the [Event](#) class.

**optionschange**

Map options changed.

**sizechange**

Map size changed. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `oldSize` - The previous size of the map.
- `newSize` - The new size of the map.

**typechange**

The map type changed. Instance of the [Event](#) class.

## Methods details

**destroy**

```
{ } destroy()
```

Destroys the map.

**getBounds**

```
{Number[][]} getBounds([options])
```

**Returns** a two-dimensional array of geocoordinates for the lower-left and upper-right corners of the map viewport.

**Parameters:**

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.useMapMargin</a>	false	Type: Boolean Whether to use offsets that were calculated in the margins manager <a href="#">map.margin.Manager</a> .

## getCenter

```
{Number[]} getCenter([options])
```

Returns geographical coordinates of the current map center.

### Parameters:

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.useMapMargin</a>	false	Type: Boolean Whether to use offsets that were calculated in the margins manager <a href="#">map.margin.Manager</a> .

## getGlobalPixelCenter

```
{Number[]} getGlobalPixelCenter([options])
```

Returns global pixel coordinates of the current map center.

### Parameters:

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.useMapMargin</a>	false	Type: Boolean Whether to use offsets that were calculated in the margins manager <a href="#">map.margin.Manager</a> .

### Example:

```
// Shifting the map 10 pixels left  
var position = map.getGlobalPixelCenter();  
map.setGlobalPixelCenter([ position[0] - 10, position[1] ]);
```

## getPanoramaManager

```
{vow.Promise.<panorama.Manager>} getPanoramaManager()
```

Returns [vow.Promise](#) which:

- will be **resolved** with: [<panorama.Manager>](#) — The panorama manager for the map;
- either **rejected** with an error.

## getType

```
{String|MapType} getType()
```

Returns the current map type.

## getZoom

```
{Number} getZoom()
```

**Returns** the current map zoom.

## panTo

```
{vow.Promise} panTo(center[, options])
```

Sets the map center. If an array of points is passed, the map will move from one point to the other.

**Returns** Promise object. If the action completed successfully, returns "resolve" without a value. If an error occurred, returns "reject" with the error description.

### Parameters:

Parameter	Default value	Description
<code>center</code> *	—	Type: Number[] Object[]  The map center or an array of points to move through sequentially.
<code>options</code>	—	Type: Object  Options.
<code>options.checkZoomRange</code>	false	Type: Boolean  Checks whether the current zoom can be set after moving the map center. If the value of this option is "true", the method is called asynchronously. A request is sent to the server, which returns the range of acceptable zoom values for the given center. After this, the specified center and appropriate zoom are applied.
<code>options.delay</code>	1000	Type: Number  The delay time between moves, in milliseconds.
<code>options.duration</code>	500	Type: Number  Animation duration, in milliseconds.

Parameter	Default value	Description
<code>options.flying</code>	true	Type: Boolean  Allow decreasing and then increasing the map zoom when moving between points. If the distance between points is less than twice the size of the map container, the map smoothly moves the center to the specified point without changing the zoom level. If the distance is more, the map applies a smaller zoom level, then moves to the destination point and applies the required zoom level (emulates flying over the map).
<code>options.safe</code>	true	Type: Boolean  Shifts the map safely. You can only use it if the "flying" option is false. If the distance between points is less than twice the size of the map container, the map smoothly moves the center to the specified point. If the distance exceeds that value, the map instantly moves to the selected point. Use this mode to conveniently move the map without displaying unloaded tiles.
<code>options.timingFunction</code>	'ease-in-out'	Type: String  Timing function. The same as the value of the CSS property <code>transition-timing-function</code> . Full list of values: <a href="http://www.w3.org/TR/css3-transitions/#transition-timing-function_tag">http://www.w3.org/TR/css3-transitions/#transition-timing-function_tag</a>
<code>options.useMapMargin</code>	false	Type: Boolean  Whether to use offsets that were calculated in the margins manager <a href="#">map.margin.Manager</a> .

\* Mandatory parameter/option.

#### Example:

```
// Flight from Kaliningrad to Vladivostok via Moscow
map.setCenter([54.704815, 20.466380], 10);
map.panTo([
  [55.751574, 37.573856],
  [43.134091, 131.928478]
]).then(function () {
  alert('Landed!');
}, function (err) {
  alert('An error occurred ' + err);
});
```

```
}, this);
```

## setBounds

```
{vow.Promise} setBounds(bounds[, options])
```

Positions the map for displaying the region that was passed.

**Returns** Promise object. If the action completed successfully, returns "resolve" without a value. If an error occurred, returns "reject" with the error description.

### Parameters:

Parameter	Default value	Description
<code>bounds</code> *	—	Type: Number[][] Boundaries of the viewport.
<code>options</code>	—	Type: Object Options.
<code>options.checkZoomRange</code>	false	Type: Boolean Checks whether the specified zoom level can be set. If the value of this option is "true", the method is called asynchronously. A request is sent to the server, which returns the range of acceptable zoom values for the given center. After this, the specified center and appropriate zoom are applied.
<code>options.duration</code>	0	Type: Number Animation duration, in milliseconds.
<code>options.preciseZoom</code>	false	Type: Boolean The ability to use fractional zoom levels.
<code>options.timingFunction</code>	'linear'	Type: String Timing function. The same as the value of the CSS property transition-timing-function. Full list of values: <a href="http://www.w3.org/TR/css3-transitions/#transition-timing-function_tag">http://www.w3.org/TR/css3-transitions/#transition-timing-function_tag</a>
<code>options.useMapMargin</code>	false	Type: Boolean Whether to use offsets that were calculated in the margins manager <code>map.margin.Manager</code> .

Parameter	Default value	Description
<code>options.zoomMargin</code>	0	Type: Number Number[]  Offset from the borders of the visible area of the map. If a single number is set, it is applied to each side. If two numbers are set, they are the horizontal and vertical margins, respectively. If an array of four numbers is set, they are the top, right, bottom, and left margins. When the "useMapMargin" option is enabled, the "zoomMargin" value is combined with the values that were calculated in the margins manager <a href="#">map.margin.Manager</a> .

\* Mandatory parameter/option.

#### Example:

```
// The new map center and zoom level are calculated based on the current
// map state.
// If the current zoomRange does not match the zoomRange for the new map center,
// grey tiles may be displayed if the viewport is very small.
// To avoid this problem, use the checkZoomRange option.
map.setBounds([[60,-40], [20,60]], {
  checkZoomRange: true,
}).then(function () {
  // Action has completed successfully.
}, function (err) {
  // Specified region could not be shown
  // ...
}, this);
```

#### setCenter

```
{vow.Promise} setCenter(center[, zoom[, options]])
```

Sets the map center and zoom level. The center is set in geographical coordinates.

**Returns** Promise object. If the action completed successfully, returns "resolve" without a value. If an error occurred, returns "reject" with the error description.

#### Parameters:

Parameter	Default value	Description
<code>center</code> *	—	Type: Number[]  Geo coordinates of the map center.
<code>zoom</code>	—	Type: Number  Map zoom level.
<code>options</code>	—	Type: Object  Options.

Parameter	Default value	Description
<code>options.checkZoomRange</code>	false	Type: Boolean  Checks whether the specified zoom level can be set. If the value of this option is "true", the method is called asynchronously. A request is sent to the server, which returns the range of acceptable zoom values for the given center. After this, the specified center and appropriate zoom are applied.
<code>options.duration</code>	0	Type: Number  Animation duration, in milliseconds.
<code>options.timingFunction</code>	'linear'	Type: String  Timing function. The same as the value of the CSS property transition-timing-function. Full list of values: <a href="http://www.w3.org/TR/css3-transitions/#transition-timing-function_tag">http://www.w3.org/TR/css3-transitions/#transition-timing-function_tag</a>
<code>options.useMapMargin</code>	false	Type: Boolean  Whether to use offsets that were calculated in the margins manager <a href="#">map.margin.Manager</a> .

\* Mandatory parameter/option.

#### Example:

```
myMap.setCenter([40, 50], 3, {
  checkZoomRange: true
});
```

#### setGlobalPixelCenter

```
{vow.Promise} setGlobalPixelCenter(globalPixelCenter[, zoom[, options]])
```

Sets the map center and zoom level. The center is set in global pixel coordinates.

**Returns** Promise object. If the action completed successfully, returns "resolve" without a value. If an error occurred, returns "reject" with the error description.

#### Parameters:

Parameter	Default value	Description
<code>globalPixelCenter</code> *	—	Type: Number[]  Pixel coordinates of the new map center.

Parameter	Default value	Description
<code>zoom</code>	—	Type: Number  Map zoom level.
<code>options</code>	—	Type: Object  Options.
<code>options.checkZoomRange</code>	false	Type: Boolean  Checks whether the specified zoom level can be set. If the value of this option is "true", the method is called asynchronously. A request is sent to the server, which returns the range of acceptable zoom values for the given center. After this, the specified center and appropriate zoom are applied.
<code>options.duration</code>	0	Type: Number  Animation duration, in milliseconds.
<code>options.timingFunction</code>	'linear'	Type: String  Timing function. The same as the value of the CSS property transition-timing-function. Full list of values: <a href="http://www.w3.org/TR/css3-transitions/#transition-timing-function_tag">http://www.w3.org/TR/css3-transitions/#transition-timing-function_tag</a>
<code>options.useMapMargin</code>	false	Type: Boolean  Whether to use offsets that were calculated in the margins manager <code>map.margin.Manager</code> .

\* Mandatory parameter/option.

#### Example:

```
// Shifting the map 10 pixels left
var position = map.getGlobalPixelCenter();
map.setGlobalPixelCenter([ position[0] - 10, position[1] ]);
```

#### setType

```
{vow.Promise} setType(type[, options])
```

Sets the map type.

**Returns** Promise object. If the action completed successfully, returns "resolve" without a value. If an error occurred, returns "reject" with the error description.

**Parameters:**

Parameter	Default value	Description
<code>type</code> *	—	Type: <code>String MapType</code>  Map type. Can be a key or an instance of the <code>MapType</code> class. List of available keys: <ul style="list-style-type: none"> <li>'yandex#map' - "Roadmap" map type.</li> <li>'yandex#satellite' - "Satellite" map type.</li> <li>'yandex#hybrid' - "Hybrid" map type.</li> </ul>
<code>options</code>	—	Type: <code>Object</code>  Map options.
<code>options.checkZoomRange</code>	<code>false</code>	Type: <code>Boolean</code>  Checks if the specified map type can be set at the specified zoom level. When set to "true", a request is sent to the server, which returns the range of acceptable zoom values for the given map type. After this, the specified center and appropriate zoom are applied.

\* Mandatory parameter/option.

#### Example:

```
map.setType('yandex#hybrid', {
  checkZoomRange: true
}).then(function () {
  // The map type has been set with the acceptable zoom level.
}, this);
```

#### setZoom

```
{vow.Promise} setZoom(zoom[, options])
```

Sets the map zoom level.

**Returns** Promise object. If the action completed successfully, returns "resolve" without a value. If an error occurred, returns "reject" with the error description.

#### Parameters:

Parameter	Default value	Description
<code>zoom</code> *	—	Type: <code>Number</code>  Map zoom level.
<code>options</code>	—	Type: <code>Object</code>  Options.

Parameter	Default value	Description
<a href="#">options.checkZoomRange</a>	false	Type: Boolean  Checks whether the specified zoom level can be set.
<a href="#">options.duration</a>	0	Type: Number  Animation duration, in milliseconds.
<a href="#">options.useMapMargin</a>	false	Type: Boolean  Whether to use offsets that were calculated in the margins manager <a href="#">map.margin.Manager</a> .

\* Mandatory parameter/option.

#### Example:

```
myMap.setZoom(4, {duration: 1000});
```

## map

### map.action

#### map.action.Continuous

Extends [IMapAction](#).

Map movement consisting of one or more steps. Intended for implementing complex map movements.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
map.action.Continuous()
```

#### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

#### Events

Name	Description
<a href="#">end</a>	Event that notifies the map that movement has finished.  Inherited from <a href="#">IMapAction</a> .

Name	Description
<a href="#">tick</a>	<p>Event that notifies the map of the next step. Contains the fields:</p> <ul style="list-style-type: none"> <li><code>globalPixelCenter</code> - The new map center, in global pixels.</li> <li><code>zoom</code> - The new map zoom.</li> <li><code>duration</code> - The time that is allowed for performing the step.</li> <li><code>timingFunction</code> - Function describing the type of movement.</li> </ul> <p>Inherited from <a href="#">IMapAction</a>.</p>

## Methods

Name	Returns	Description
<a href="#">begin(mapActionManager)</a>		<p>Starts the movement to be performed by the map. This method is called automatically by the map movement manager. From the moment when <a href="#">IMapAction.begin</a> is called, the movement manager listens for <a href="#">IMapAction.event:tick</a> and <a href="#">IMapAction.event:end</a> and executes them.</p> <p>Inherited from <a href="#">IMapAction</a>.</p>
<a href="#">end()</a>		<p>Stops movement.</p> <p>Inherited from <a href="#">IMapAction</a>.</p>
<a href="#">isActive()</a>	Boolean	<p>Checks whether map movement is being performed at this moment.</p>
<a href="#">tick(tick)</a>	<a href="#">map.action.Continuous</a>	<p>Performs a single step of the map movement.</p>

## Methods details

### isActive

```
{Boolean} isActive()
```

Checks whether map movement is being performed at this moment.

**Returns** true if the movement is currently being performed by the map, otherwise false.

### tick

```
{map.action.Continuous} tick(tick)
```

Performs a single step of the map movement.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>tick *</code>	—	Type: Object  Movement parameters.
<code>tick.duration</code>	0	Type: Number  Duration of making the move, in milliseconds.
<code>tick.globalPixelCenter</code>	—	Type: Number[]  The new map center in global pixels. One of the parameters must be set: either <code>pixelOffset</code> , or <code>globalPixelCenter</code> .
<code>tick.pixelOffset</code>	—	Type: Number[]  The offset in pixels relative to the previous center. One of the parameters must be set: either <code>pixelOffset</code> , or <code>globalPixelCenter</code> .
<code>tick.timingFunction</code>	'linear'	Type: String  Timing function.
<code>tick.zoom</code>	—	Type: Number  The new map zoom. If omitted, the map zoom does not change.

\* Mandatory parameter/option.

### map.action.Manager

Extends [IEventEmitter](#).

Map actions manager. Makes complex movements possible on the map and ensures that complex movements do not overlap each other. Every map already has its own actions manager, available as [Map.action](#). Don't create new instances of this class unless necessary.

See [Map.action](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
map.action.Manager(map)
```

### Parameters:

Parameter	Default value	Description
<code>map *</code>	—	Type: <a href="#">Map</a>  <a href="#">Map</a> .

\* Mandatory parameter/option.

**Example:**

```
// Creating a complex movement: every 100 ms, the map
// center shifts a random amount.

// Creating an instance of the complex movement
var action = new ymaps.map.action.Continuous();
// Executing it on the map
myMap.action.execute(action);

// Recalling the pixel map center and zoom level
var center = myMap.getGlobalPixelCenter();
var zoom = myMap.getZoom();
// Generate a random shift every 100 milliseconds
var interval = window.setInterval(function () {
  center[0] += Math.round(Math.random() * 100) - 50;
  center[1] += Math.round(Math.random() * 100) - 50;
  // Generating a new shift in the map
  action.tick({
    globalPixelCenter: center,
    zoom: zoom
  });
}, 100);

// As soon as the user moves the map, our movement
// stops being executed and the "end" event occurs.
var listener = action.events.once('end', function () {
  listener.removeAll();
  window.clearInterval(interval);
});
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

**Events**

Name	Description
<a href="#">begin</a>	Event that occurs when an action has started. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>action - Action that has started.</li> </ul>
<a href="#">break</a>	Event that occurs when an action step was prematurely stopped (for example, because another action or another step of the action was performed). Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>action - An action.</li> </ul>
<a href="#">end</a>	Event that occurs when an action is stopped. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>action - Action that has stopped.</li> </ul>
<a href="#">tick</a>	Event that occurs when the next step of an action begins to be performed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>action - The action being performed at this moment.</li> <li>tick - Description of an action step in the form of an object with the fields "globalPixelCenter", "zoom", "duration" and "timingFunction".</li> </ul>
<a href="#">tickcomplete</a>	Event that occurs when an action step has been completed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>action - The action being performed at this moment.</li> <li>tick - Description of an action step in the form of an object with the fields "globalPixelCenter", "zoom", "duration" and "timingFunction".</li> </ul>

**Methods**

Name	Returns	Description
<a href="#">breakTick()</a>		Interrupts an action step.
<a href="#">execute(action)</a>		Starts an action on the map. If a different movement is being performed on the map at this time, it is stopped (the "end" method is called). The new movement is started using a "begin" method call.
<a href="#">getCurrentState()</a>	Object	Checks the map state at the time of smooth movement.
<a href="#">getMap()</a>	Map	Returns reference to the map.
<a href="#">setCorrection(userFunction)</a>		Used for making user adjustments to complex movements on the map. When the adjustment is finished, the corrected values must be returned.
<a href="#">stop()</a>		Stops an action being performed on the map.

**Events details****begin**

Event that occurs when an action has started. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- action - Action that has started.

**break**

Event that occurs when an action step was prematurely stopped (for example, because another action or another step of the action was performed). Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- action - An action.

**end**

Event that occurs when an action is stopped. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- action - Action that has stopped.

**tick**

Event that occurs when the next step of an action begins to be performed. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- action - The action being performed at this moment.
- tick - Description of an action step in the form of an object with the fields "globalPixelCenter", "zoom", "duration" and "timingFunction".

**Example:**

```
// Tracks all map movement, even user dragging
```

```
// and smooth zooming
myMap.action.events.add('tick', function (e) {
  var tick = e.get('tick');
  console.log('Now the map is moving to the point (' +

myMap.options.get('projection').fromGlobalPixels(tick.globalPixelCenter, tick.zoom).join(',') +
  ') during ' + e.get('tick').duration + ' milliseconds');
});
```

### tickcomplete

Event that occurs when an action step has been completed. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- action - The action being performed at this moment.
- tick - Description of an action step in the form of an object with the fields "globalPixelCenter", "zoom", "duration" and "timingFunction".

### Methods details

#### breakTick

```
{ } breakTick()
```

Interrupts an action step.

#### execute

```
{ } execute(action)
```

Starts an action on the map. If a different movement is being performed on the map at this time, it is stopped (the "end" method is called). The new movement is started using a "begin" method call.

#### Parameters:

Parameter	Default value	Description
<a href="#">action</a> *	—	Type: <a href="#">IMapAction</a> Action.

\* Mandatory parameter/option.

#### getCurrentState

```
{Object} getCurrentState()
```

Checks the map state at the time of smooth movement.

**Returns** object with the fields: isTicking - Whether a step of smooth movement is being performed. tickProgress - Which part of the current step has been completed. zoom - The map zoom during the current step. globalPixelCenter - The map center in global pixels during the current step.

#### Example:

```
// Logs the current map center.
// Even works during smooth zooming or
// while the user is dragging the map.
window.setInterval(function () {
  console.log(myMap.action.getCurrentState().center.join(', '));
}, 100);
```

#### getMap

```
{Map} getMap()
```

**Returns** reference to the map.

### setCorrection

```
{ } setCorrection(userFunction)
```

Used for making user adjustments to complex movements on the map. When the adjustment is finished, the corrected values must be returned.

#### Parameters:

Parameter	Default value	Description
<a href="#">userFunction</a> *	—	Type: Function  Custom function for adjusting steps.

\* Mandatory parameter/option.

#### Example:

```
// Making it impossible for the user to drag the map center
// outside of the Moscow Ring Road.
var mkad = [
  [55.785017, 37.841576],
  [55.861979, 37.765992],
  [55.898533, 37.635961],
  [55.888897, 37.48861],
  [55.83251, 37.395275],
  [55.744789, 37.370248],
  [55.660424, 37.434424],
  [55.5922, 37.526366],
  [55.574019, 37.683167],
  [55.62913, 37.802473],
  [55.712203, 37.837121]
];
var mkadPolygon = new ymaps.Polygon([mkad], {}, {
  fillColor: '#FFFF00',
  opacity: .4
});
myMap.geoObjects.add(mkadPolygon);
myMap.action.setCorrection(function (tick) {
  var projection = myMap.options.get('projection');
  var tickCenter = projection.fromGlobalPixels(tick.globalPixelCenter, tick.zoom);
  // If the map center is not in our area.
  if (!mkadPolygon.geometry.contains(tickCenter)) {
    tick.globalPixelCenter = projection.toGlobalPixels(
      mkadPolygon.geometry.getClosest(tickCenter).position,
      tick.zoom
    );
    tick.duration = 0;
  }
  return tick;
});
```

### stop

```
{ } stop()
```

Stops an action being performed on the map.

### map.action.Single

Extends [IMapAction](#).

Simple map movement. The movement is made immediately after it is passed to `map.action.Manager`.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
map.action.Single(tick)
```

Creates a simple (single step) map movement.

**Parameters:**

Parameter	Default value	Description
<code>tick *</code>	—	Type: Object  Movement parameters.
<code>tick.callback</code>	—	Type: Function  Function that will be called after performing the action. Accepts an error or null as a parameter, if the action was not successful.
<code>tick.center</code>	—	Type: Number[]  The new map center in geo coordinates.
<code>tick.checkZoomRange</code>	false	Type: Boolean  Flag showing whether the new map zoom needs to be checked. If the flag is set to true, the range of allowable zoom levels at the new point will be requested before performing the action. If the specified zoom does not fall within the allowable values, it will be corrected. In addition, the value of the new map center in global pixel coordinates will be changed.
<code>tick.duration</code>	0	Type: Number  Duration of making the move, in milliseconds.
<code>tick.globalPixelCenter</code>	—	Type: Number[]  The new map center in global pixels. When the "center" and "globalPixelCenter" parameters are set simultaneously, the "center" parameter is ignored.
<code>tick.timingFunction</code>	'linear'	Type: String  Timing function.
<code>tick.zoom</code>	—	Type: Number  The new map zoom.

\* Mandatory parameter/option.

**Example:**

```
var myCallback = function(err) {
  if (err) {
    throw err;
  }
},
myAction = new ymaps.map.action.Single({
  center: [0, 0],
  zoom: 4,
  duration: 1000,
```

```

    timingFunction: 'ease-in',
    checkZoomRange: true,
    callback: myCallback
  });

// The action is performed immediately after calling "execute".
myMap.action.execute(myAction);

```

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

## Events

Name	Description
<a href="#">end</a>	Event that notifies the map that movement has finished. Inherited from <a href="#">IMapAction</a> .
<a href="#">tick</a>	Event that notifies the map of the next step. Contains the fields: <ul style="list-style-type: none"> <li><code>globalPixelCenter</code> - The new map center, in global pixels.</li> <li><code>zoom</code> - The new map zoom.</li> <li><code>duration</code> - The time that is allowed for performing the step.</li> <li><code>timingFunction</code> - Function describing the type of movement.</li> </ul> Inherited from <a href="#">IMapAction</a> .

## Methods

Name	Returns	Description
<a href="#">begin(mapActionManager)</a>		Starts the movement to be performed by the map. This method is called automatically by the map movement manager. From the moment when <a href="#">IMapAction.begin</a> is called, the movement manager listens for <a href="#">IMapAction.event:tick</a> and <a href="#">IMapAction.event:end</a> and executes them. Inherited from <a href="#">IMapAction</a> .
<a href="#">end()</a>		Stops movement. Inherited from <a href="#">IMapAction</a> .
<a href="#">isActive()</a>	Boolean	Checks whether map movement is being performed at this moment.

## Methods details

### isActive

```
{Boolean} isActive()
```

Checks whether map movement is being performed at this moment.

**Returns** true if the movement is currently being performed by the map, otherwise false.

## map.addon

### map.addon.balloon

**Note:** The constructor of the `map.addon.balloon` class is hidden, as this class is not intended for autonomous initialization.

Static object.

The module that makes it possible to use a balloon for the map. Adds the [IBalloonOwner](#) interface to the map ([Map](#)). When enabling `package.full` (the standard set of modules), it is available by default. If [Map](#) is enabled separately, this module must be explicitly specified in the loader. If `map.addon.balloon` is enabled separately after creating [Map](#), the [IBalloonOwner](#) interface won't be added. Then in order to initialize the balloon manager, you will need to use the `map.addon.balloon#get` method.

#### Methods

#### Methods

Name	Returns	Description
<a href="#">get(map)</a>	<a href="#">IPopupManager</a>	Returns map balloon manager.

#### Methods details

#### get

```
{IPopupManager} get (map)
```

**Returns** map balloon manager.

#### Parameters:

Parameter	Default value	Description
<a href="#">map</a> *	—	Type: <a href="#">Map</a> Map

\* Mandatory parameter/option.

#### Example:

```
ymaps.map.addon.balloon.get (myMap)
```

### map.addon.hint

**Note:** The constructor of the `map.addon.hint` class is hidden, as this class is not intended for autonomous initialization.

Static object.

The module that makes it possible to use a hint for the map. Adds the [IHintOwner](#) interface to the map ([Map](#)). When enabling `package.full` (the standard set of modules), it is available by default. If [Map](#) is enabled separately, this module must be explicitly specified in the loader. If `map.addon.hint` is enabled separately after creating [Map](#), the [IHintOwner](#) interface won't be added. Then in order to initialize the balloon manager, you will need to use the `map.addon.hint#get` method.

#### Methods

**Methods**

Name	Returns	Description
<a href="#">get(map)</a>	<a href="#">IPopupManager</a>	Returns map hint manager.

**Methods details****get**

```
{IPopupManager} get (map)
```

Returns map hint manager.

**Parameters:**

Parameter	Default value	Description
<a href="#">map</a> *	—	Type: <a href="#">Map</a> Map

\* Mandatory parameter/option.

**Example:**

```
ymaps.map.addon.hint.get(myMap)
```

**map.Balloon**

Extends [IBalloonManager](#), [IBalloonSharingManager](#).

Map balloon manager. Each map already has its own balloon manager, available as `myMap.balloon`. Only one balloon controlled by this manager can be open on the map at a time. Don't create new instances of this class unless necessary.

See [Balloon Map.balloon](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
map.Balloon(map)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">map</a> *	—	Type: <a href="#">Map</a> Reference to a map object.

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

## Events

Name	Description
<a href="#">autopanbegin</a>	<p>Start of automatic shifting of the map center initiated by the <code>autoPan</code> method. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>target</code> - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> <p>Inherited from <a href="#">IBalloonManager</a>.</p>
<a href="#">autopanend</a>	<p>End of automatic shifting of the map center initiated by the <code>autoPan</code> method. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>target</code> - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> <p>Inherited from <a href="#">IBalloonManager</a>.</p>
<a href="#">beforeuserclose</a>	<p>The event which precedes <a href="#">Balloon.event:userclose</a>. Allows you to cancel the user's action by calling the <code>preventDefault</code> method. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>target</code> - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> <p>Inherited from <a href="#">IBalloonManager</a>.</p>
<a href="#">close</a>	<p>Closing the info object. Names of fields available via <a href="#">Event.get</a>:</p> <ul style="list-style-type: none"> <li><code>target</code> - Reference to the object where the closing occurred.</li> </ul> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">open</a>	<p>Opening the info object. Names of fields available via <a href="#">Event.get</a>:</p> <ul style="list-style-type: none"> <li><code>target</code> - Reference to the object where the opening occurred.</li> </ul> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">userclose</a>	<p>Balloon closed by the user. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>target</code> - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> <p>Inherited from <a href="#">IBalloonManager</a>.</p>

## Methods

Name	Returns	Description
<a href="#">autoPan()</a>	<a href="#">vow.Promise</a>	<p>Moves the map so that the balloon is visible.</p> <p>Inherited from <a href="#">IBalloonManager</a>.</p>
<a href="#">close([force])</a>	<a href="#">vow.Promise</a>	<p>Closes the info object.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>
<a href="#">destroy()</a>		<p>Disables the info object manager.</p> <p>Inherited from <a href="#">IPopupManager</a>.</p>

Name	Returns	Description
<a href="#">getData()</a>	Object null	Returns the data of the info object or 'null'.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOptions()</a>	<a href="#">IOptionManager</a>  null	Returns the options manager or 'null'.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object to return the overlay.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	Returns the overlay, if one exists.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">getPosition()</a>	Number[][] null	Returns the coordinates of the info object or 'null'.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">isOpen()</a>	Boolean	Returns the info object state: open/closed.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">open([position[, data[, options]])</a>	<a href="#">vow.Promise</a>	Opens the info object at the specified position.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">setData(data)</a>	<a href="#">vow.Promise</a>	Defines new data for the info object.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">setOptions(options)</a>	<a href="#">vow.Promise</a>	Defines new options for the info object.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">setPosition(position)</a>	<a href="#">vow.Promise</a>	Specifies a new position for the info object.  Inherited from <a href="#">IPopupManager</a> .

## map.behavior

### map.behavior.Manager

Extends [ICustomizable](#), [IEventEmitter](#), [IParentOnMap](#).

Map behaviors manager. Allows to enable and disable behaviors. Each map already has its own behavior manager, available as `map.behaviors`. Don't instantiate new instances of this class unless necessary

See [Map.behaviors](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

## Constructor

```
map.behavior.Manager(map[, behaviors[, options]])
```

### Parameters:

Parameter	Default value	Description
<code>map</code> *	—	Type: <a href="#">Map</a>  Map.
<code>behaviors</code>	—	Type: <code>String String[]</code>  List of map behaviors that are immediately enabled when a map is created. By default - "drag", "dblClickZoom" and "rightMouseButtonMagnifier" for desktop browsers; "drag", "dblClickZoom" and "multiTouch" for mobile browsers.  Acceptable key values: <ul style="list-style-type: none"> <li>"default" - Shortcut for enabling/disabling default map behaviors.</li> <li>"drag" - Dragging the map when the left mouse button is held down, or by a single touch <a href="#">behavior.Drag</a>.</li> <li>"scrollZoom" - Changing the zoom with the mouse wheel <a href="#">behavior.ScrollZoom</a>.</li> <li>"dblClickZoom" - Zooming the map on a double click <a href="#">behavior.DblClickZoom</a>.</li> <li>"multiTouch" - Zooming the map with a multi touch (i.e. on a touch screen) <a href="#">behavior.MultiTouch</a>.</li> <li>"rightMouseButtonMagnifier" - Magnifying the area that is selected using the right mouse button (for desktop browsers only), <a href="#">behavior.RightMouseButtonMagnifier</a>.</li> <li>"leftMouseButtonMagnifier" - Magnifying the area selected by the left mouse button or a single touch, <a href="#">behavior.LeftMouseButtonMagnifier</a>.</li> <li>"ruler" - Measuring distance <a href="#">behavior.Ruler</a>.</li> <li>"routeEditor" - Route editor <a href="#">behavior.RouteEditor</a>.</li> </ul> You can add and remove behavior classes via the behaviors storage <a href="#">behavior.storage</a> .

Parameter	Default value	Description
<a href="#">options</a>	—	<p>Type: Object</p> <p>Behavior options. The following options can be set:</p> <ul style="list-style-type: none"> <li>Options for the <a href="#">behavior.Drag</a> behavior with the <code>drag</code> prefix.</li> <li>Options for the <a href="#">behavior.ScrollZoom</a> behavior with the <code>scrollZoom</code> prefix.</li> <li>Options for the <a href="#">behavior.DblClickZoom</a> behavior with the <code>dblClickZoom</code> prefix.</li> <li>Options for the <a href="#">behavior.MultiTouch</a> behavior with the <code>multiTouch</code> prefix.</li> <li>Options for the <a href="#">magnifier.RightMouseButtonMagnifier</a> behavior with the <code>rightMouseButtonMagnifier</code> prefix.</li> <li>Options for the <a href="#">behavior.LeftMouseButtonMagnifier</a> behavior with the <code>leftMouseButtonMagnifier</code> prefix.</li> <li>Options for the <a href="#">behavior.Ruler</a> behavior with the <code>ruler</code> prefix.</li> </ul>

\* Mandatory parameter/option.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

### Events

Name	Description
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"> <li><code>oldMap</code> - Old map.</li> <li><code>newMap</code> - New map.</li> </ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">disable(behaviors)</a>	<a href="#">map.behavior.Manager</a>	Disables behaviors on the map.
<a href="#">enable(behaviors)</a>	<a href="#">map.behavior.Manager</a>	Enables behaviors on the map.
<a href="#">get(behaviorName)</a>	<a href="#">IBehavior</a>	Returns instance of the behavior by the key.
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">isEnabled(behaviorName)</a>	Boolean	Checks whether a behavior is currently enabled.

**Methods details****disable**

```
{map.behavior.Manager} disable(behaviors)
```

Disables behaviors on the map.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<a href="#">behaviors</a> *	—	Type: <a href="#">String String[]</a>  List of behaviors that can be disabled.

\* Mandatory parameter/option.

**Example:**

```
myMap.behaviors.disable('drag');
```

**enable**

```
{map.behavior.Manager} enable(behaviors)
```

Enables behaviors on the map.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<a href="#">behaviors</a> *	—	Type: <a href="#">String String[]</a>  List of behaviors that can be enabled.

\* Mandatory parameter/option.

**Example:**

```
myMap.behaviors.enable(['ruler', 'multiTouch']);
```

**get**

```
{IBehavior} get(behaviorName)
```

**Returns** instance of the behavior by the key.

**Parameters:**

Parameter	Default value	Description
<a href="#">behaviorName</a> *	—	Type: String Name of the behavior.

\* Mandatory parameter/option.

**Example:**

```
myMap.behaviors.get('drag');
```

**isEnabled**

```
{Boolean} isEnabled(behaviorName)
```

Checks whether a behavior is currently enabled.

**Returns** true if the behavior is enabled, otherwise false.

**Parameters:**

Parameter	Default value	Description
<a href="#">behaviorName</a> *	—	Type: String Behavior ID.

\* Mandatory parameter/option.

**Example:**

```
// If the "drag" behavior is disabled, we enable it
if (!(myMap.behaviors.isEnabled('drag'))) {
  myMap.behaviors.enable('drag');
}
```

**map.Container**

Extends [IDomEventEmitter](#).

Map container manager. Each map already has its own container manager, available as `map.container`. Don't instantiate new instances of this class unless necessary.

See [Map.container](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
map.Container(parentElement)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">parentElement</a> *	—	Type: String HTMLElement  HTML element (or its ID) where the map will be created.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IDomEventEmitter</a> .

## Events

Name	Description
<a href="#">beforefullscreenexit</a>	The event preceding the "fullscreenexit" event. If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent fullscreenexit event will be canceled. Instance of the <a href="#">Event</a> class.
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">fullscreenenter</a>	The map switched to fullscreen mode. Instance of the <a href="#">Event</a> class.
<a href="#">fullscreenexit</a>	The map exited full-screen mode. Instance of the <a href="#">Event</a> class.
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseleave</a>	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">sizechange</a>	<p>Change to the size of the map container. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>oldSize</code>: <a href="#">Number[]</a>;</li><li>• <code>newSize</code>: <a href="#">Number[]</a>;</li><li>• <code>oldOffset</code>: <a href="#">Number[]</a>;</li><li>• <code>newOffset</code>: <a href="#">Number[]</a>;</li><li>• <code>preservePixelPosition</code>: <a href="#">Boolean</a>.</li></ul>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">enterFullscreen()</a>		Allows you to switch the map to full-screen mode.
<a href="#">exitFullscreen()</a>		Allows you to take the map out of full-screen mode.
<a href="#">fitToViewport</a> ( <a href="#">[[preservePixelPosition]]</a> )		Called when the size of the map container is changed so that the map applies the new size.
<a href="#">getElement()</a>	HTMLElement	Returns map HTML element.
<a href="#">getOffset()</a>	Number[]	Returns the shift of the map container in pixels, relative to the upper-left corner of the document.
<a href="#">getParentElement()</a>	HTMLElement	Returns custom HTML element which contains the created map.
<a href="#">getSize()</a>	Number[]	Returns size of the map container, in pixels.
<a href="#">isFullscreen()</a>	Boolean	Returns indicates whether the map is in full-screen mode.

## Events details

### beforefullscreenexit

The event preceding the "fullscreenexit" event. If the [Event.preventDefault](#) method is called for this event, a subsequent fullscreenexit event will be canceled. Instance of the [Event](#) class.

### fullscreenenter

The map switched to fullscreen mode. Instance of the [Event](#) class.

### fullscreenexit

The map exited full-screen mode. Instance of the [Event](#) class.

### sizechange

Change to the size of the map container. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `oldSize`: Number[];
- `newSize`: Number[];
- `oldOffset`: Number[];
- `newOffset`: Number[];
- `preservePixelPosition`: Boolean.

## Methods details

### enterFullscreen

```
{ } enterFullscreen()
```

Allows you to switch the map to full-screen mode.

### exitFullscreen

```
{ } exitFullscreen()
```

Allows you to take the map out of full-screen mode.

### fitToViewport

```
{ } fitToViewport([preservePixelPosition])
```

Called when the size of the map container is changed so that the map applies the new size.

#### Parameters:

Parameter	Default value	Description
<a href="#">preservePixelPosition</a>	—	Type: Boolean Save the location of the map center.

#### Example:

```
// Changing the dimensions of the map container  
map.container.getElement().style.width = '300px';  
// Initializing size recalculation  
map.container.fitToViewport();
```

### getElement

```
{HTMLInputElement} getElement()
```

Returns map HTML element.

### getOffset

```
{Number[]} getOffset()
```

Returns the shift of the map container in pixels, relative to the upper-left corner of the document.

### getParentElement

```
{HTMLInputElement} getParentElement()
```

Returns custom HTML element which contains the created map.

### getSize

```
{Number[]} getSize()
```

Returns size of the map container, in pixels.

**isFullscreen**

```
{Boolean} isFullscreen()
```

**Returns** indicates whether the map is in full-screen mode.

**map.Converter**

Class for converting global pixel coordinates of a point (calculated from the upper-left corner of the world) to local coordinates (calculated from the upper-left corner of the window) and back. Each map already has its own converter, available as `map.converter`. Don't instantiate new instances of this class unless necessary.

See [Map.converter](#)

[Constructor](#) | [Methods](#)

**Constructor**

```
map.Converter(map)
```

**Parameters:**

Parameter	Default value	Description
<code>map</code> *	—	Type: <a href="#">Map</a> Reference to the map.

\* Mandatory parameter/option.

**Methods**

Name	Returns	Description
<code>globalToPage(globalPixelPoint)</code>	<code>Number[]</code>	Converts global pixel coordinates of a point to local coordinates.
<code>pageToGlobal(pagePixelPoint)</code>	<code>Number[]</code>	Converts local pixel coordinates of a point to global coordinates.

**Methods details****globalToPage**

```
{Number[]} globalToPage(globalPixelPoint)
```

Converts global pixel coordinates of a point to local coordinates.

**Returns** the converted coordinates.

**Parameters:**

Parameter	Default value	Description
<code>globalPixelPoint</code> *	—	Type: <code>Number[]</code> Pixel coordinates of a point that need to be converted.

\* Mandatory parameter/option.

**Example:**

```
// Converting geographical coordinates to browser window pixels
var projection = map.options.get('projection');
console.log(map.converter.globalToPage(
  projection.toGlobalPixels(
    // geographical coordinates
    [55, 37],
    map.getZoom()
  )
));
```

**pageToGlobal**

```
{Number[]} pageToGlobal(pagePixelPoint)
```

Converts local pixel coordinates of a point to global coordinates.

**Returns** the converted coordinates.

**Parameters:**

Parameter	Default value	Description
<a href="#">pagePixelPoint</a> *	—	Type: <code>Number[]</code>  Pixel coordinates of a point that need to be converted.

\* Mandatory parameter/option.

**Example:**

```
// Converting mouse cursor coordinates to geocoordinates
var projection = map.options.get('projection');
$('#map').bind('click', function (e) {
  console.log(projection.fromGlobalPixels(
    map.converter.pageToGlobal([e.pageX, e.pageY]), map.getZoom()
  ).join(' '));
});
```

**map.Copyrights**

Manager for copyright information on the map. Each map already has its own copyright information manager, available as `map.copyrights`. Don't instantiate new instances of this class unless necessary.

See [Map.copyrights](#)

[Constructor](#) | [Events](#) | [Methods](#)

**Constructor**

```
map.Copyrights(map)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">map</a> *	—	Type: <code>Map</code>  <code>Map</code> .

\* Mandatory parameter/option.

**Example:**

```
// Adding static information about copyright info to the map
var accessor = map.copyrights.add('&copy; Gerardus Mercator');
// ...
// Removing information about copyrights
```

```
accessor.remove();
```

## Events

Name	Description
<a href="#">change</a>	Event for changes to copyright information placed on the map. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>oldCopyrights - Old array of copyright information.</li> <li>newCopyrights - New array of copyright information.</li> </ul>

## Methods

Name	Returns	Description
<a href="#">add(customCopyrights)</a>	<a href="#">ICopyrightsAccessor</a>	Adds static copyright information to the map (independent of the current center and zoom level).
<a href="#">addProvider(provider)</a>	<a href="#">map.Copyrights</a>	Adds a new provider of copyright information.
<a href="#">get([point[, zoom]])</a>	<a href="#">vow.Promise</a>	Determines copyright information at the specified point.
<a href="#">getPromoLink()</a>	String	Returns external link from the "Open in Yandex.Maps" block.
<a href="#">removeProvider(provider)</a>	<a href="#">map.Copyrights</a>	Removes a provider of copyright information.

## Events details

### change

Event for changes to copyright information placed on the map. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- oldCopyrights - Old array of copyright information.
- newCopyrights - New array of copyright information.

## Methods details

### add

```
{ICopyrightsAccessor} add(customCopyrights)
```

Adds static copyright information to the map (independent of the current center and zoom level).

**Returns** an object for managing the added information.

**Parameters:**

Parameter	Default value	Description
<code>customCopyrights</code> *	—	Type: String HTMLElement String[] HTMLElement[]  Copyright information in string format, a DOM element, or an array of strings/DOM elements.

\* Mandatory parameter/option.

### addProvider

```
{map.Copyrights} addProvider(provider)
```

Adds a new provider of copyright information.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>provider</code> *	—	Type: <code>ICopyrightsProvider</code>  Provider.

\* Mandatory parameter/option.

#### Example:

```
// Creating a dynamic provider of copyright information
// that will get up-to-date information about copyrights for
// a particular point on the map from a remote server.
var myProvider = {
  getCopyrights: function (center, zoom) {
    var deferred = ymaps.vow.defer();
    $.ajax('http://some.server/copyrights/?ll=' + center.join(',') + '&z=' + zoom, {
      // The server must return an array of strings
      success: function (res) {
        deferred.resolve(res);
      }
    });
    return deferred.promise();
  }
};
// Adding a provider to the map; now when the center or zoom changes,
// the map copyright information manager will
// request updated information from the remote server and display
// the received data automatically.
map.copyrights.addProvider(myProvider);
```

### get

```
{vow.Promise} get([point[, zoom]])
```

Determines copyright information at the specified point.

**Returns** a Promise object that will be resolved and will pass an array of strings or DOM elements as a parameter.

#### Parameters:

Parameter	Default value	Description
<a href="#">point</a>	—	Type: Number[]  The point (in geographical coordinates) that copyright information needs to be determined for. If omitted, the current map center is used.
<a href="#">zoom</a>	—	Type: Number  The zoom level that copyright information needs to be determined for. If omitted, the current map zoom level is used.

### getPromoLink

```
{String} getPromoLink()
```

**Returns** external link from the "Open in Yandex.Maps" block.

### removeProvider

```
{map.Copyrights} removeProvider(provider)
```

Removes a provider of copyright information.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<a href="#">provider</a> *	—	Type: <a href="#">ICopyrightsProvider</a>  Provider.

\* Mandatory parameter/option.

## map.GeoObjects

Extends [IGeoObjectCollection](#).

Collection of map geo objects. Each map already has its own geo objects collection, available as `map.geoObjects`. Don't instantiate new instances of this class unless necessary.

See [Map.geoObjects](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
map.GeoObjects(map[, options])
```

#### Parameters:

Parameter	Default value	Description
<a href="#">map</a> *	—	Type: <a href="#">Map</a>  Map.

Parameter	Default value	Description
<a href="#">options</a>	—	<p>Type: Object</p> <p>Options of a collection of geo objects. The <code>map.geoObjects</code> options can be used to make settings for geo objects that have been added to the map:</p> <ul style="list-style-type: none"> <li>Options for <a href="#">clusterers</a> with the <code>clusterer</code> prefix.</li> <li>Options for <a href="#">clusters</a> with the <code>cluster</code> prefix.</li> </ul>

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	<p>Event manager.</p> <p>Inherited from <a href="#">IEventEmitter</a>.</p>
<a href="#">options</a>	<a href="#">IOptionManager</a>	<p>Options manager.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>

## Events

Name	Description
<a href="#">add</a>	<p>A child geo object has been added (inserted). Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>index</code>: Integer - Index of the added geo object.</li> <li><code>child</code>: <a href="#">IGeoObject</a> - Reference to the added geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObjectCollection</a>.</p>
<a href="#">boundschange</a>	<p>Change to coordinates of the geographical area that spans the collection and its child geo objects. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IGeoObjectCollection</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"> <li><code>oldMap</code> - Old map.</li> <li><code>newMap</code> - New map.</li> </ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">pixelboundschange</a>	<p>Change to pixel coordinates of the area that includes the collection and its child geo objects. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IGeoObjectCollection</a>.</p>
<a href="#">remove</a>	<p>A child geo object has been removed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>index</code>: Integer - Index of the deleted geo object.</li> <li><code>child</code>: <a href="#">IGeoObject</a> - Reference to the deleted geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObjectCollection</a>.</p>

Name	Description
<a href="#">set</a>	<p>A new child geo object has been added to the collection. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>index: Integer - Index of the geo object.</li> <li>child: <a href="#">IGeoObject</a> - Reference to the new geo object.</li> <li>prevChild: <a href="#">IGeoObject</a> - Reference to the previous value for this index.</li> </ul> <p>Inherited from <a href="#">IGeoObjectCollection</a>.</p>

## Methods

Name	Returns	Description
<a href="#">add(child[, index])</a>	<a href="#">map.GeoObjects</a>	Adds (inserts) a child geo object to the collection.
<a href="#">each(callback[, context])</a>		Calls a handler function for each child geo object.
<a href="#">get(index)</a>	<a href="#">IGeoObject</a>	Returns a child geo object with the specified index.
<a href="#">getBounds()</a>	Number[][] null	Returns geographical coordinates of the area that covers the collection and its child geo objects.
<a href="#">getIterator()</a>	<a href="#">Iterator</a>	Returns iterator for the collection.
<a href="#">getLength()</a>	Integer	Returns length of the collection.
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getPixelBounds()</a>	Number[][] null	Returns global pixel coordinates of the area that spans the collection and its child geo objects.
<a href="#">indexOf(object)</a>	Integer	Returns index of the child geo object. If the geo object cannot be found in the collection, -1 is returned.
<a href="#">remove(child)</a>	<a href="#">map.GeoObjects</a>	Removes a child geo object from the collection.
<a href="#">removeAll()</a>	<a href="#">map.GeoObjects</a>	Clears the collection.
<a href="#">set(index, child)</a>	<a href="#">map.GeoObjects</a>	Adds a new child geo object to the collection.
<a href="#">splice(index, number)</a>	<a href="#">GeoObjectCollection</a>	Removes geo objects from the collection. If necessary, puts other objects in their place. Objects that will be added in place of the deleted ones are passed as additional parameters (after the "number" parameter).

## Methods details

### add

```
{map.GeoObjects} add(child[, index])
```

Adds (inserts) a child geo object to the collection.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>child</code> *	—	Type: <a href="#">IGeoObject</a> Child object.
<code>index</code>	—	Type: Integer The index where the new object is added. By default, the object is added to the end of the collection.

\* Mandatory parameter/option.

### each

```
{ } each(callback[, context])
```

Calls a handler function for each child geo object.

#### Parameters:

Parameter	Default value	Description
<code>callback</code> *	—	Type: Function Handler function.
<code>context</code>	—	Type: Object Context for the handler function.

\* Mandatory parameter/option.

#### Example:

```
// Displaying a geo object's index in a collection for its icon contents.
myGeoObjects.events.add(["add", "remove", "set"], function () {
  this.each(function (el, i) {
    el.properties.set("iconContent", i);
  })
}, myGeoObjects);
```

### get

```
{IGeoObject} get(index)
```

**Returns** a child geo object with the specified index.

#### Parameters:

Parameter	Default value	Description
<code>index *</code>	—	Type: Integer Index.

\* Mandatory parameter/option.

### getBounds

```
{Number[][]|null} getBounds()
```

**Returns** geographical coordinates of the area that covers the collection and its child geo objects.

### getIterator

```
{IIterator} getIterator()
```

**Returns** iterator for the collection.

### getLength

```
{Integer} getLength()
```

**Returns** length of the collection.

### getPixelBounds

```
{Number[][]|null} getPixelBounds()
```

**Returns** global pixel coordinates of the area that spans the collection and its child geo objects.

### indexOf

```
{Integer} indexOf(object)
```

**Returns** index of the child geo object. If the geo object cannot be found in the collection, -1 is returned.

**Parameters:**

Parameter	Default value	Description
<code>object *</code>	—	Type: Object Child object.

\* Mandatory parameter/option.

### remove

```
{map.GeoObjects} remove(child)
```

Removes a child geo object from the collection.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>child *</code>	—	Type: <a href="#">IGeoObject</a> Geo object being removed.

\* Mandatory parameter/option.

#### Example:

```
// When a geo object is clicked, we remove it from the collection.
myGeoObjects.events.add("click", function (e) {
  if (e.get("target").getParent() == this) {
    this.remove(e.get("target"));
  }
}, myGeoObjects);
```

#### removeAll

```
{map.GeoObjects} removeAll()
```

Clears the collection.

**Returns** self-reference.

#### set

```
{map.GeoObjects} set(index, child)
```

Adds a new child geo object to the collection.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>index *</code>	—	Type: Integer Index.
<code>child *</code>	—	Type: <a href="#">IGeoObject</a> Child object.

\* Mandatory parameter/option.

#### splice

```
{GeoObjectCollection} splice(index, number)
```

Removes geo objects from the collection. If necessary, puts other objects in their place. Objects that will be added in place of the deleted ones are passed as additional parameters (after the "number" parameter).

**Returns** collection of deleted geo objects.

#### Parameters:

Parameter	Default value	Description
<code>index *</code>	—	Type: Integer Index of the geo object to start deletion from.

Parameter	Default value	Description
<code>number</code> *	—	Type: Integer  The number of geo objects to be deleted.

\* Mandatory parameter/option.

#### Example:

```
// Removes the second object.
myGeoObjects.splice(1, 1);
// Puts a new "obj" object in the second position.
myGeoObjects.splice(1, 0, obj);
// Replaces the second object with the new "obj" object.
myGeoObjects.splice(1, 1, obj);
```

## map.Hint

Extends [IHintManager](#), [IHintSharingManager](#).

Map hint manager. Each map already has its own hint manager, available as `myMap.hint`. Only one hint, controlled by this manager, can be open on the map at a time. Don't create new instances of this class unless necessary.

See [Hint Map.hint](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
map.Hint(map)
```

### Parameters:

Parameter	Default value	Description
<code>map</code> *	—	Type: <a href="#">Map</a>  Reference to a map object.

\* Mandatory parameter/option.

### Fields

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

### Events

Name	Description
<code>close</code>	Closing the info object. Names of fields available via <a href="#">Event.get</a> : <ul style="list-style-type: none"> <li><code>target</code> - Reference to the object where the closing occurred.</li> </ul> Inherited from <a href="#">IPopupManager</a> .
<code>open</code>	Opening the info object. Names of fields available via <a href="#">Event.get</a> : <ul style="list-style-type: none"> <li><code>target</code> - Reference to the object where the opening occurred.</li> </ul> Inherited from <a href="#">IPopupManager</a> .

## Methods

Name	Returns	Description
<a href="#">close</a> ([force])	<a href="#">vow.Promise</a>	Closes the info object. Inherited from <a href="#">IPopupManager</a> .
<a href="#">destroy</a> ()		Disables the info object manager. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getData</a> ()	Object null	Returns the data of the info object or 'null'. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOptions</a> ()	<a href="#">IOptionManager</a>  null	Returns the options manager or 'null'. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOverlay</a> ()	<a href="#">vow.Promise</a>	Returns the promise object to return the overlay. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOverlaySync</a> ()	<a href="#">IOverlay</a>  null	Returns the overlay, if one exists. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getPosition</a> ()	Number[] null	Returns the coordinates of the info object or 'null'. Inherited from <a href="#">IPopupManager</a> .
<a href="#">isOpen</a> ()	Boolean	Returns the info object state: open/closed. Inherited from <a href="#">IPopupManager</a> .
<a href="#">open</a> ([position[, data[, options]])	<a href="#">vow.Promise</a>	Opens the info object at the specified position. Inherited from <a href="#">IPopupManager</a> .
<a href="#">setData</a> (data)	<a href="#">vow.Promise</a>	Defines new data for the info object. Inherited from <a href="#">IPopupManager</a> .
<a href="#">setOptions</a> (options)	<a href="#">vow.Promise</a>	Defines new options for the info object. Inherited from <a href="#">IPopupManager</a> .

Name	Returns	Description
<a href="#">setPosition(position)</a>	<a href="#">vow.Promise</a>	Specifies a new position for the info object.  Inherited from <a href="#">IPopupManager</a> .

## map.layer

### map.layer.Manager

Extends [ILayer](#), [IMapObjectCollection](#).

Map layers manager.

See [Map.layers](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
map.layer.Manager(map[, options])
```

Creates an instance of the class for working with map layers. Each map already has its own layer manager, available as `map.layers`. Don't instantiate new instances of this class unless necessary.

### Parameters:

Parameter	Default value	Description
<a href="#">map</a> *	—	Type: <a href="#">Map</a>  Map
<a href="#">options</a>	—	Type: Object  Map layer options. The <code>map.layers</code> options can be used to make settings for layers that have been added to the map. Options for hotspot layers are set using the "hotspotLayer" prefix.
<a href="#">options.trafficImageZIndex</a>	201	Type: Number  The z-index of the traffic picture layer.
<a href="#">options.trafficInfoZIndex</a>	1	Type: Number  Priority of the hotspot layer of info points.
<a href="#">options.trafficJamZIndex</a>	0	Type: Number  Priority of the hotspot layer for traffic jams.

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

**Events**

Name	Description
<a href="#">add</a>	A child object was added. Inherited from <a href="#">ICollection</a> .
<a href="#">brightnesschange</a>	Layer brightness change event. Inherited from <a href="#">ILayer</a> .
<a href="#">copyrightschange</a>	Event for changes to available copyright information. Inherited from <a href="#">ILayer</a> .
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> Inherited from <a href="#">IParentOnMap</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .
<a href="#">remove</a>	A child object was deleted. Inherited from <a href="#">ICollection</a> .
<a href="#">tileloadchange</a>	Tile upload status change event. Data fields: <ul style="list-style-type: none"> <li>readyTileNumber - Number of ready tiles. A tile is considered ready when it is downloaded and rendered. Type: Number.</li> <li>totalTileNumber - Total number of visible tiles. Type: Number.</li> </ul> Inherited from <a href="#">ILayer</a> .
<a href="#">zoomrangechange</a>	Event for changes to available information about the zoom level range. Inherited from <a href="#">ILayer</a> .

## Methods

Name	Returns	Description
<a href="#">add(object)</a>	<a href="#">ICollection</a>	Adds a child object to the collection.  Inherited from <a href="#">ICollection</a> .
<a href="#">each(callback[, context])</a>		Iterates through all the items in the collection and calls a handler function for each of them.
<a href="#">getBrightness()</a>	Number	Optional method.  Inherited from <a href="#">ILayer</a> .
<a href="#">getCopyrights(coords, zoom)</a>	<a href="#">vow.Promise</a>	Optional method. Requests information about copyrights at the specified point with the specified zoom.  Inherited from <a href="#">ILayer</a> .
<a href="#">getIterator()</a>	<a href="#">Iterator</a>	Returns iterator for the collection.  Inherited from <a href="#">ICollection</a> .
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map.  Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IChildOnMap</a> .
<a href="#">getZoomRange(point)</a>	<a href="#">vow.Promise</a>	Optional method. Checks the available range of zoom levels at the specified point. If there is data, the returned promise object will be resolved and will pass as a result an array of two numbers - the minimum and maximum zoom level available at the point. If there is no data, the promise is rejected with an error.  Inherited from <a href="#">ILayer</a> .
<a href="#">remove(object)</a>	<a href="#">ICollection</a>	Removes a child object from the collection.  Inherited from <a href="#">ICollection</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .

## Methods details

### each

```
{ } each(callback[, context])
```

Iterates through all the items in the collection and calls a handler function for each of them.

#### Parameters:

Parameter	Default value	Description
<code>callback</code> *	—	Type: Function Handler function.
<code>context</code>	—	Type: Object Context for the function.

\* Mandatory parameter/option.

## map.margin

### map.margin.Accessor

An object that provides access to the rectangular area in the margins manager.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
map.margin.Accessor(screenArea)
```

#### Parameters:

Parameter	Default value	Description
<code>screenArea</code> *	—	Type: Object  The rectangular area, which is set in screen coordinates. This area is defined as an object containing information about the margins from the map edges (left, top, right, bottom) and the dimensions of the area (width, height). The values can be set as percentages of the width/height of the map container.  Don't instantiate new instances of this class unless necessary.

\* Mandatory parameter/option.

#### Fields

Name	Type	Description
<code>events</code>		Event manager.

**Events**

Name	Description
<a href="#">change</a>	The rectangular area was changed.
<a href="#">remove</a>	The accessor was deleted from the margins manager.

**Methods**

Name	Returns	Description
<a href="#">getArea()</a>	Object	Returns the rectangular area.
<a href="#">remove()</a>	<a href="#">map.margin.Accessor</a>	Deletes the rectangular area from the margins manager.
<a href="#">setArea(screenArea)</a>	<a href="#">map.margin.Accessor</a>	Returns self-reference.

**Fields details****events**

```
events
```

Event manager.

**Events details****change**

The rectangular area was changed.

**remove**

The accessor was deleted from the margins manager.

**Methods details****getArea**

```
{Object} getArea()
```

**Returns** the rectangular area.

**remove**

```
{map.margin.Accessor} remove()
```

Deletes the rectangular area from the margins manager.

**Returns** self-reference.

**setArea**

```
{map.margin.Accessor} setArea(screenArea)
```

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<a href="#">screenArea</a> *	—	Type: Object  The rectangular area, which is set in screen coordinates. This area is defined as an object containing information about the margins from the map edges (left, top, right, bottom) and the dimensions of the area (width, height). The values can be set as percentages of the width/height of the map container.

\* Mandatory parameter/option.

#### Example:

```
accessor.setArea({
  top: 50,
  left: 50,
  width: 100,
  height: 100
});
```

### map.margin.Manager

Extends [IEventEmitter](#).

Map margins manager.

A manager for calculating the optimal margins from the edge of the map container.

As input, the manager accepts a definition of rectangular areas in screen coordinates, which designate the occupied areas of the map container.

Margins can be used when setting the current map viewport, in order to get the best display of data on the map. The data on the map never ends up under the occupied areas of the map container. [Map#setBound](#).

Don't instantiate new instances of this class unless necessary.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
map.margin.Manager()
```

#### Parameters:

Parameter	Default value	Description
<a href="#">map</a> *	—	Type: <a href="#">Map</a>

\* Mandatory parameter/option.

#### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

## Events

Name	Description
<a href="#">change</a>	Changes to map margins.

## Methods

Name	Returns	Description
<a href="#">addArea(screenArea)</a>	<a href="#">map.margin.Accessor</a>	Adding a new rectangular area.
<a href="#">destroy()</a>	<a href="#">map.margin.Manager</a>	Destroying the margins manager.
<a href="#">getMargin()</a>	Number[]	Returns the current margins from the map edges. Values in the array are ordered as: upper margin, right margin, lower margin, left margin.
<a href="#">getOffset()</a>	Number[]	Returns the difference (in pixels) between the geometric center of the map (without margins) and the logical center (with consideration for the margins).
<a href="#">setDefaultMargin(margin)</a>		Sets the default margins from the map edges.

## Events details

### change

Changes to map margins.

## Methods details

### addArea

```
{map.margin.Accessor} addArea(screenArea)
```

Adding a new rectangular area.

**Returns** the object that provides access to the added area. To delete the added rectangular area, you must use this object.

**Parameters:**

Parameter	Default value	Description
<code>screenArea</code> *	—	Type: Object  The rectangular area, which is set in local coordinates. This area is defined as an object containing information about the margins from the map edges (left, top, right, bottom) and the dimensions of the area (width, height). The values can be set as percentages of the width/height of the map container.

\* Mandatory parameter/option.

### Examples:

#### 1.

```
// Offset from the upper-left corner.
map.margin.addArea({
  left: 0,
  top: 0,
  width: 78,
  height: 101
});
console.log(map.margin.getMargin()); // [0, 0, 0, 78]
```

#### 2.

```
// Offset from the lower-right corner.
map.margin.addArea({
  right: 50,
  bottom: 50,
  width: 26,
  height: 25
});
console.log(map.margin.getMargin()); // [0, 0, 75, 0]
```

#### 3.

```
// Setting the margins as a percentage.
map.margin.addArea({
  left: 50,
  bottom: '1%',
  width: 20,
  height: 20
});
console.log(map.margin.getMargin()); // [0, 0, 26, 0]
```

#### 4.

```
// Setting the sizes as a percentage.
map.margin.addArea({
  left: 0,
  top: 50,
  width: '100%',
  height: 25
});
console.log(map.margin.getMargin()); // [75, 0, 0, 0]
```

#### 5.

```
// Setting multiple areas.
map.margin.addArea({
  top: 10,
  left: 10,
  width: 20,
  height: 20
});
map.margin.addArea({
  top: 20,
  right: 40,
  width: 100,
```

```

    height: 100
  });
  map.margin.addArea({
    bottom: 20,
    left: 30,
    width: 120,
    height: 30
  });
  console.log(map.margin.getMargin()); // [120, 0, 50, 0]

```

## destroy

```
{map.margin.Manager} destroy()
```

Destroying the margins manager.

**Returns** self-reference.

## getMargin

```
{Number[]} getMargin()
```

**Returns** the current margins from the map edges. Values in the array are ordered as: upper margin, right margin, lower margin, left margin.

## getOffset

```
{Number[]} getOffset()
```

**Returns** the difference (in pixels) between the geometric center of the map (without margins) and the logical center (with consideration for the margins).

## setDefaultMargin

```
{ } setDefaultMargin(margin)
```

Sets the default margins from the map edges.

### Parameters:

Parameter	Default value	Description
<code>margin</code> *	—	Type: Number Number[]  Margin values in the form of one, two, or four numbers (similar to setting margins in CSS).

\* Mandatory parameter/option.

## map.pane

### map.pane.Manager

Map pane manager. Each map already has its own pane manager, available as `map.panes`. Don't create new instances of this class unless necessary. The list of default keys for map panes and their `zIndex` values:

- 'ground': [pane.MovablePane](#) (`zIndex`: 100) - The lowest pane, intended for the map's base layer.
- 'areas': [pane.MovablePane](#) (`zIndex`: 200) - Pane for objects with an area, like a polygon.
- 'shadows': [pane.MovablePane](#) (`zIndex`: 300) - Pane for the shadows of map objects that are above it.
- 'places': [pane.MovablePane](#) (`zIndex`: 400) - Pane for point objects, such as placemarks.
- 'events': [pane.EventsPane](#) (`zIndex`: 500) - Pane for listening to map events.

- 'overlaps': [pane.MovablePane](#) (zIndex: 600) - Pane for objects that don't require hotspots to make them interactive.
- 'balloon': [pane.MovablePane](#) (zIndex: 700) - Balloon pane.
- 'outerBalloon': [pane.MovablePane](#) (zIndex: 800) - External balloon pane.
- 'controls': [pane.StaticPane](#) (zIndex: 900) - Map controls pane.
- 'hint': [pane.StaticPane](#) (zIndex: 1100) - Hint pane.
- 'outerHint': [pane.StaticPane](#) (zIndex: 1200) - External hint pane.

See [Map.panes](#)

[Constructor](#) | [Methods](#)

### Constructor

```
map.pane.Manager (map)
```

#### Parameters:

Parameter	Default value	Description
<a href="#">map</a> *	—	Type: <a href="#">Map</a> Map.

\* Mandatory parameter/option.

#### Example:

```
// Adding a watermark on top of the map container.
// To do this, we will change the event container's background.
map.panes.get('events').getElement().style.backgroundImage = 'my/background/image';
```

### Methods

Name	Returns	Description
<a href="#">append(key, pane)</a>		Adds a new pane to the map. The key of the pane being added must be unique within the current set of keys for the map panes.
<a href="#">destroy()</a>		Destructor.
<a href="#">get(key)</a>	<a href="#">IPane</a>  null	Returns either the map pane with the given key, or null, if the requested pane is not available on the map.
<a href="#">getLower()</a>	String	Takes the keys of map panes as arguments and returns the key of the lowest pane in the received set. If no key was specified, the search is performed on the entire set of map panes.

Name	Returns	Description
<code>getUpper()</code>	String	Takes the keys of map panes as arguments and returns the key of the top pane in the received set. If no key was specified, the search is performed on the entire set of map panes.
<code>insertBefore(key, pane, referenceKey)</code>		Inserts a new pane in front of another map pane. The key of the pane being added must be unique within the current set of keys for the map panes.
<code>remove(pane)</code>		Deletes a pane from the map.

## Methods details

### append

```
{ } append (key, pane)
```

Adds a new pane to the map. The key of the pane being added must be unique within the current set of keys for the map panes.

#### Parameters:

Parameter	Default value	Description
<code>key *</code>	—	Type: String The key of the pane being added.
<code>pane *</code>	—	Type: <code>IPane</code> The pane being added.

\* Mandatory parameter/option.

### destroy

```
{ } destroy()
```

Destructor.

### get

```
{IPane|null} get (key)
```

**Returns** either the map pane with the given key, or null, if the requested pane is not available on the map.

#### Parameters:

Parameter	Default value	Description
<code>key *</code>	—	Type: String The pane key.

\* Mandatory parameter/option.

### getLower

```
{String} getLower()
```

Takes the keys of map panes as arguments and returns the key of the lowest pane in the received set. If no key was specified, the search is performed on the entire set of map panes.

**Returns** the key for the lowest map pane.

### getUpper

```
{String} getUpper()
```

Takes the keys of map panes as arguments and returns the key of the top pane in the received set. If no key was specified, the search is performed on the entire set of map panes.

**Returns** the key for the uppermost map pane.

### insertBefore

```
{ } insertBefore(key, pane, referenceKey)
```

Inserts a new pane in front of another map pane. The key of the pane being added must be unique within the current set of keys for the map panes.

#### Parameters:

Parameter	Default value	Description
<code>key</code> *	—	Type: String The key of the pane being added.
<code>pane</code> *	—	Type: <code>IPane</code> The pane being added.
<code>referenceKey</code> *	—	Type: String The key of the pane to insert the new pane in front of.

\* Mandatory parameter/option.

### remove

```
{ } remove (pane)
```

Deletes a pane from the map.

#### Parameters:

Parameter	Default value	Description
<code>pane</code> *	—	Type: <code>IPane</code> The pane being deleted.

\* Mandatory parameter/option.

## map.ZoomRange

Extends [IEventEmitter](#).

Map zoom level manager. Each map already has its own zoom level manager, available as `map.zoomRange`. Don't instantiate new instances of this class unless necessary.

See [Map.zoomRange](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
map.ZoomRange(map, constraints)
```

### Parameters:

Parameter	Default value	Description
<code>map</code> *	—	Type: <a href="#">Map</a> Map.
<code>constraints</code> *	—	Type: <a href="#">Number[]</a> An array containing the minimum and maximum map zooms.

\* Mandatory parameter/option.

### Fields

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

### Events

Name	Description
<code>change</code>	Changes occurred in the map zoom range.

### Methods

Name	Returns	Description
<code>get([coords])</code>	<a href="#">vow.Promise</a>	Checks the available range of zoom levels at the specified point.
<code>getCurrent()</code>	<a href="#">Number[]</a>	Returns the current (last received) values of the minimum and maximum map zoom levels.

### Events details

#### change

Changes occurred in the map zoom range.

## Methods details

### get

```
{vow.Promise} get([coords])
```

Checks the available range of zoom levels at the specified point.

**Returns** a Promise object, which will be resolved and will get an array of two numbers as a parameter - the maximum and minimum zoom at the given point.

#### Parameters:

Parameter	Default value	Description
<code>coords</code>	—	Type: Number[]  Coordinates of a point. If omitted, the current map center is used.

#### Example:

```
// Finding the coordinates of the Yandex office and showing it on the map
// at the maximum possible zoom level.
ymaps.geocode('Moscow, Lev Tolstoy street, 16').then(function (res) {
  var coords = res.geoObjects.get(0).geometry.getCoordinates();
  map.zoomRange.get(coords).then(function (range) {
    map.setCenter(coords, range[1]);
  });
});
```

### getCurrent

```
{Number[]} getCurrent()
```

**Returns** the current (last received) values of the minimum and maximum map zoom levels.

## MapEvent

Extends [Event](#).

Object that describes an event that took place on the map. Names of fields that are available via the [Event.get](#) method:

- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- Coordinates of the event in global pixels from the upper-left corner of the page (also available by the name "position").
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

[Constructor](#) | [Methods](#)

#### Constructor

```
MapEvent(originalEvent[, sourceEvent])
```

#### Parameters:

Parameter	Default value	Description
<a href="#">originalEvent</a> *	—	Type: Object  Data associated with the event. It should contain the "map" field referencing the map on which the event has occurred.
<a href="#">sourceEvent</a>	—	Type: <a href="#">IEvent</a>  Source event.

\* Mandatory parameter/option.

#### Example:

```
// Opening a balloon at the point where the map was clicked
map.events.add('click', function (e) {
  map.balloon.open(e.get('coords'), 'Click!');
});
```

#### Methods

Name	Returns	Description
<a href="#">allowMapEvent()</a>		Allows the propagation of the event to the map.  Inherited from <a href="#">IEvent</a> .
<a href="#">callMethod(name)</a>	Object	Calls the specified method. The operation is equivalent to searching fields via "get" and making a call that passes <a href="#">originalEvent</a> as context. All arguments after the first one are passed as parameters to the method being called.  Inherited from <a href="#">Event</a> .
<a href="#">get(name)</a>	Object	Returns the field value from <a href="#">originalEvent</a> . <a href="#">originalEvent</a> always has the following fields: <ul style="list-style-type: none"> <li>• type - String event type.</li> <li>• target - Reference to the object that generated the event.</li> </ul> Inherited from <a href="#">Event</a> .
<a href="#">getSourceEvent()</a>	<a href="#">IEvent</a>  null	Returns source event.  Inherited from <a href="#">IEvent</a> .
<a href="#">isDefaultPrevented()</a>	Boolean	Checks whether the default reaction to the event is canceled in the Yandex.Maps API event system.  Inherited from <a href="#">Event</a> .

Name	Returns	Description
<a href="#">isImmediatePropagationStopped()</a>	Boolean	Checks whether event propagation is stopped in the Yandex.Maps API event system.  Inherited from <a href="#">Event</a> .
<a href="#">isMapEventAllowed()</a>	Boolean	Returns true if the map event is enabled.  Inherited from <a href="#">IEvent</a> .
<a href="#">isPropagationStopped()</a>	Boolean	Checks whether event propagation up the hierarchy of objects and collections is stopped in the Yandex.Maps API event system.  Inherited from <a href="#">Event</a> .
<a href="#">preventDefault()</a>		Cancels the default reaction to an event within the Yandex.Maps API event system. Calling this method does not affect propagation of the DOM source event (if there is one) through the DOM tree.  Inherited from <a href="#">Event</a> .
<a href="#">stopImmediatePropagation()</a>		Stops event propagation in the Yandex.Maps API event system. Calling this method does not affect propagation of the DOM source event (if there is one) through the DOM tree.  Inherited from <a href="#">Event</a> .
<a href="#">stopPropagation()</a>		Stops event propagation up the hierarchy of objects and collections in the Yandex.Maps API event system. Calling this method does not affect propagation of the DOM source event (if there is one) through the DOM tree.  Inherited from <a href="#">Event</a> .

## MapType

Map type.

[Constructor](#) | [Methods](#)

### Constructor

```
MapType(name, layers)
```

Creates an instance of the map type.

**Parameters:**

Parameter	Default value	Description
<code>name</code> *	—	Type: String Type name.
<code>layers</code> *	—	Type: <code>Function[] String[]</code> Array containing constructors for layers or keys.

\* Mandatory parameter/option.

**Example:**

```
// Creating a custom map type that consists of satellite images from MapQuest
// with the Yandex.Hybrid overlay.

// Class of MapQuest tiles
var MQLayer = function () {
  var layer = new ymaps.Layer('http://oatile%d.mqcdn.com/naip//%z/%x/%y.jpg');
  // Copyrights
  layer.getCopyrights = function () {
    return ymaps.vow.resolve('Data, imagery and map information provided by MapQuest, Open Street Map and
    contributors, CC-BY-SA');
  };
  // Range of available zoom levels
  layer.getZoomRange = function () {
    return ymaps.vow.resolve([0, 18]);
  };

  return layer;
};

// Adding a layer with the key
ymaps.layer.storage.add('mq#aerial', MQLayer);
// Creating a map type that consists of the layers 'mq#aerial' and 'yandex#skeleton'
var myMapType = new ymaps.MapType('MQ + Ya', ['mq#aerial', 'yandex#skeleton']);
// Adding it to the map type storage
ymaps.mapType.storage.add('mq_ya#hybrid', myMapType);
// Now we can set our map type for any map
map.setType('mq_ya#hybrid');
```

**Methods**

Name	Returns	Description
<code>getLayers()</code>	<code>Function[] String[]</code>	Returns a list of layers for the given map type - an array of constructors or keys for layers.
<code>getName()</code>	String	Returns name of the map type.

**Methods details****getLayers**

```
{Function[]|String[]} getLayers()
```

**Returns** a list of layers for the given map type - an array of constructors or keys for layers.

**getName**

```
{String} getName()
```

**Returns** name of the map type.

**Parameters:**

Parameter	Default value	Description
<code>map</code> *	—	Type:

\* Mandatory parameter/option.

## mapType

### mapType.storage

Static object.

Instance of [util.Storage](#)

Storage for map types.

#### Methods

#### Example:

```
// Adding a custom map type

// Let's say we have our own map layer
var myLayer = function () {
    return new ymaps.Layer('http://some.server/?%c');
}
// Adding it to the layer storage
ymaps.layer.storage.add('my#layer', myLayer);
// Creating our own map type, consisting of a single layer
var myType = new ymaps.MapType('My map type', ['my#layer']);
// Adding it to the map type storage
ymaps.mapType.storage.add('my#mapType', myType);
// Now we can assign our map type to a map
map.setType('my#type');
```

#### Methods

Name	Returns	Description
<code>add(key, object)</code>	<a href="#">util.Storage</a>	Adds an object to storage.
<code>get(key)</code>	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
<code>remove(key)</code>	<a href="#">util.Storage</a>	Deletes the "key: value" pair from storage.

## meta

Static object.

Information about the API.

#### Fields

**Fields**

Name	Type	Description
<a href="#">coordinatesOrder</a>	String	The order of coordinates used in the API. Possible values: <ul style="list-style-type: none"><li>latlong — [latitude, longitude]</li><li>longlat — [longitude, latitude]</li></ul> Set by the "coordorder" GET parameter when enabling the API. <a href="#">More information about API setup parameters</a>
<a href="#">countryCode</a>	String	Two-letter country code. Returned in <a href="#">ISO 3166-1</a> format. Set by the "lang" GET parameter when enabling the API. <a href="#">More information about API setup parameters</a>
<a href="#">languageCode</a>	String	Two-letter language code. Returned in <a href="#">ISO 639-1</a> format. Set by the "lang" GET parameter when enabling the API. <a href="#">More information about API setup parameters</a>
<a href="#">mode</a>	String	Yandex.Maps API mode. Possible values: <ul style="list-style-type: none"><li>release</li><li>debug</li></ul> Set by the "mode" GET parameter when enabling the API. <a href="#">More information about API setup parameters</a>
<a href="#">ns</a>	Object	Link to the Yandex.Maps API namespace. It has a value independent of the "ns" parameter value when enabling the API.
<a href="#">version</a>	String	Version of the Yandex.Maps API.

**Fields details****coordinatesOrder**

```
{String} coordinatesOrder
```

The order of coordinates used in the API. Possible values:

- latlong — [latitude, longitude]
- longlat — [longitude, latitude]

Set by the "coordorder" GET parameter when enabling the API.

[More information about API setup parameters](#)

**countryCode**

```
{String} countryCode
```

Two-letter country code. Returned in [ISO 3166-1](#) format.

Set by the "lang" GET parameter when enabling the API.

[More information about API setup parameters](#)

### languageCode

```
{String} languageCode
```

Two-letter language code. Returned in [ISO 639-1](#) format.

Set by the "lang" GET parameter when enabling the API.

[More information about API setup parameters](#)

### mode

```
{String} mode
```

Yandex.Maps API mode. Possible values:

- release
- debug

Set by the "mode" GET parameter when enabling the API.

[More information about API setup parameters](#)

### ns

```
{Object} ns
```

Link to the Yandex.Maps API namespace.

It has a value independent of the "ns" parameter value when enabling the API.

### version

```
{String} version
```

Version of the Yandex.Maps API.

## modules

Static object.

The module system that the Yandex.Maps API is based on.

The Yandex.Maps API consists of a large number of interconnected modules. A module is a programming unit. For example, a class, a specific realization of a class, a static object, or a function. The module system guarantees that when initializing a particular module, all the modules it needs will already be initialized.

The module system provides asynchronous access, since it may be necessary to load missing modules.

You can add your own modules to the module system.

### modules.define

Static function.

Defining a module in the module system.

**Note:** We recommend creating your custom modules in your own namespace, to avoid accidentally replacing the modules that are necessary for the API to work.

Returns self-reference.

```
{ modules } modules.define(module[, depends, resolveCallback[, context]])
```

### Parameters:

Parameter	Default value	Description
<code>module</code> *	—	Type: String  Module name.
<code>depends</code>	—	Type: String[]  Array of names of necessary modules. This argument may be omitted.
<code>resolveCallback</code> *	—	Type: Function  Function that defines the module. The first argument in <code>resolveCallback</code> will be a provide function, to which you will need to pass a module. The provide function call can be delayed. Subsequent arguments are the modules specified in the dependencies. The order of the modules will match the order in the <code>depends</code> array. If a module cannot be resolved, the module system must be notified. This can be done by passing a second argument to the provide function. The second argument will be passed to <code>errorCallback</code> and promise as an error in the module request. As a result, the module can be requested again.
<code>context</code>	—	Type: Object  Context for the function.

\* Mandatory parameter/option.

### Examples:

#### 1.

```
// Defining a custom module. 'plugin.*' is a custom namespace.
ymaps.modules.define('CustomModule', function (provide) {
  var CustomModule = function (defValue) {
    this.field = defValue;
  };
  provide(CustomModule);
});
// Requesting modules
ymaps.modules.require(['CustomModule'])
  .spread(
    function (CustomModule) {
      // ...
    },
    this
  );
```

#### 2.

```
// Defining a custom asynchronous module.
ymaps.modules.define('CustomAsyncModule', function (provide) {
  // For the module to work, we must load an external script.
  $.getScript("ajax/test.js")
    .done(function( script, textStatus ) {
      function CustomAsyncModule () {
        // ...
      }
    });
});
```

```

    }
    // Calling the provide function after loading the script.
    provide(CustomAsyncModule);
  });
});
// Requesting modules
ymaps.modules.require(['CustomAsyncModule'])
  .spread(
    function (CustomAsyncModule) {
      // ...
    },
    this
  );

```

**3.**

```

// Creating a custom asynchronous module with consideration for an error.
ymaps.modules.define('plugin.CustomModule', function (provide) {
  $.getScript( "ajax/test.js" )
    .done(function( script, textStatus ) {
      // ... Processing successful script loading.
      provide({ a: 1 });
    })
    .fail(function( jqxhr, settings, exception ) {
      // Notifying the module system that the module can't be prepared right now.
      provide(null, new Error('Error when loading'));
    });
});

// Requesting modules with error handling.
ymaps.modules.require(['plugin.CustomModule'])
  .spread(
    function (CustomModule) {
      // ...
    },
    function (error) {
      console.log(error.messsage); // "Error when loading".
      // Code for handling the module loading error. You can request the module one more time.
    },
    this
  );

```

**4.**

```

// Defining a custom module with dependencies.
ymaps.modules.define('CustomLayoutModule', [
  'templateLayoutFactory',
  'layout.storage'
], function (provide, templateLayoutFactory, layoutStorage) {
  var customLayoutClass = templateLayoutFactory.createClass(
    '<div>My Layout</div>',
    {
      build: function () {
        customLayoutClass.superclass.build.call(this);
        // ...
      },
      clear: function () {
        // ...
        customLayoutClass.superclass.clear.call(this);
      }
    }
  );
  layoutStorage.add('customLayout', customLayoutClass);
  provide(customLayoutClass);
});
// Requesting a custom layout
ymaps.modules.require(['CustomLayoutModule'])
  .spread(
    function (CustomLayoutModule) {
      // ...
    },
    this
  );

```

**modules.isDefined**

Static function.

Checking the module's availability by name.

**Returns** true if the module is defined, false if not.

```
{ Boolean } modules.isDefined(module)
```

**Parameters:**

Parameter	Default value	Description
<code>module</code> *	—	Type: String Module

\* Mandatory parameter/option.

#### Example:

```
if (ymaps.modules.isDefined('plugin.CustomModule')) {
  ymaps.modules.require(['plugin.CustomModule'])
    .spread(
      function (CustomModule) {
        // ...
      }
    );
}
```

## modules.require

Static function.

Request to get modules.

**Returns** a promise object that is confirmed by the requested modules. Or it is rejected, if an error occurred. For example, one of the requested modules is missing in the module system.

```
{ vow.Promise } modules.require(modules[, successCallback[, errorCallback[, context]])
```

#### Parameters:

Parameter	Default value	Description
<code>modules</code> *	—	Type: String String[] Module name or array of module names.
<code>successCallback</code>	—	Type: Function The function that is called after receiving all the modules. The requested entities will be passed to the function as arguments. The order of the arguments will match the order in the modules array.
<code>errorCallback</code>	—	Type: Function The function that will be called in case of an error.
<code>context</code>	—	Type: Object The execution context of the callback function.

\* Mandatory parameter/option.

#### Examples:

##### 1.

```
// Requesting the 'Map' and 'Placemark' modules using a promise object.
ymaps.modules.require(['Map', 'Placemark'])
  .spread(
    function (Map, Placemark) {
      var myMap = new Map("map", {
```

```

        center: [55.72, 37.64],
        zoom: 5
    });
    myMap.geoObjects.add(
        new Placemark(myMap.getCenter())
    );
},
function (error) {
    // Error handling.
},
this
);

```

**2.**

```

// Requesting the 'Map' module.
ymaps.modules.require('Map')
    .spread(
        function (Map) {
            var myMap = new Map("map", {
                center: [55.72, 37.64],
                zoom: 5
            });
        },
        this
    );

```

## Monitor

Object that tracks changes to certain data fields in the specified data manager. It can also be used for tracking changes to options.

[Constructor](#) | [Methods](#)

### Constructor

```
Monitor(dataManager)
```

### Parameters:

Parameter	Default value	Description
<code>dataManager</code> *	—	Type: <a href="#">IDataManager</a>   <a href="#">OptionManager</a> Data manager.

\* Mandatory parameter/option.

### Example:

```

// Tracking changes to placemark options
var placemark = new ymaps.Placemark([0, 0]);
var optionMonitor = new ymaps.Monitor(placemark.options);
optionMonitor.add("cursor", function (newValue) {
    alert("cursor: " + newValue);
});
myMap.geoObjects.add(placemark);
// Outputs the string "cursor: arrow".
myMap.options.set({
    geoObjectCursor: "arrow"
});

```

### Methods

Name	Returns	Description
<code>add(name, changeCallback[, context[, params]])</code>	<a href="#">Monitor</a>	Enables monitoring particular fields or a group of data fields.

Name	Returns	Description
<code>forceChange()</code>	Monitor	Initializes checking for changes to values of the monitored data fields.
<code>get(name)</code>	Object	Returns the current value of one of the monitored data fields.
<code>remove(name)</code>	Monitor	Disables monitoring particular fields or a group of data fields.
<code>removeAll()</code>	Monitor	Disables monitoring for all data fields.

## Methods details

### add

```
{Monitor} add(name, changeCallback[, context[, params]])
```

Enables monitoring particular fields or a group of data fields.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>name *</code>	—	Type: String String[]  Name or array of names of data fields that monitoring is being set up for.
<code>changeCallback *</code>	—	Type: Function  Handler for changes to data fields, or one of the data fields from a group.
<code>context</code>	—	Type: Object  Context for the handler of data changes and for options handlers.
<code>params</code>	—	Type: Object  Optional parameters.
<code>params.compareCallback</code>	—	Type: Function  Handler that compares the old and new values of data fields. Takes two arguments: the old value and the new value. Lower priority relative to handlers set using the compareCallbacks parameter.

Parameter	Default value	Description
<a href="#">params.compareCallbacks</a>	—	Type: Object  Hash in the format {data field name: reference to handler}. This parameter is for setting individual handlers for comparing values for various data fields in a group.
<a href="#">params.defaultValue</a>	—	Type: Object  The default value that is used if the data field is not defined.
<a href="#">params.defaultValues</a>	—	Type: Object  Hash in the format {data field name: default value}. This parameter is for setting individual default values for various data fields in a group.
<a href="#">params.resolveCallback</a>	—	Type: Function  Handler that allows the data field value. Takes two arguments; the data field name and the reference to the data manager. Lower priority relative to handlers set using the resolveCallbacks parameter.
<a href="#">params.resolveCallbacks</a>	—	Type: Object  Hash in the format {data field name: reference to handler}. This parameter is for setting individual handlers for allowing values for various data fields in a group.

\* Mandatory parameter/option.

### forceChange

```
{Monitor} forceChange()
```

Initializes checking for changes to values of the monitored data fields.

**Returns** self-reference.

### get

```
{Object} get (name)
```

**Returns** the current value of one of the monitored data fields.

**Parameters:**

Parameter	Default value	Description
<code>name *</code>	—	Type: String Data field name.

\* Mandatory parameter/option.

### remove

```
{Monitor} remove(name)
```

Disables monitoring particular fields or a group of data fields.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>name *</code>	—	Type: String String[] Name or array of names of data fields that monitoring is being disabled for.

\* Mandatory parameter/option.

### removeAll

```
{Monitor} removeAll()
```

Disables monitoring for all data fields.

**Returns** self-reference.

## multiRouter

### multiRouter.bicycle

#### multiRouter.bicycle.Path

**Note:** The constructor of the `multiRouter.bicycle.Path` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeoObject](#).

Representation of a path on a multi-stop bicycle route. A single route can contain several paths, and each path connects two waypoints.

[Fields](#) | [Events](#) | [Methods](#)

Creates a representation to display a path of the multi-stop bicycle route.

#### Fields

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .

Name	Type	Description
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">model</a>	<a href="#">multiRouter.bicycle.PathModel</a>	Data model for the multi-stop route's path.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">IDataManager</a>	Geo object data. Inherited from <a href="#">IGeoObject</a> .
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

## Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometrychange</a>	Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li> </ul> Inherited from <a href="#">IGeoObject</a> .
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> Inherited from <a href="#">IParentOnMap</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>

Name	Description
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .
<a href="#">propertieschange</a>	Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> Inherited from <a href="#">IGeoObject</a> .
<a href="#">update</a>	Updating the path rendering. Instance of the <a href="#">Event</a> class.
<a href="#">wheel</a>	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IChildOnMap</a> .

## Fields details

### model

```
{multiRouter.bicycle.PathModel} model
```

Data model for the multi-stop route's path.

### Events details

#### update

Updating the path rendering. Instance of the [Event](#) class.

### multiRouter.bicycle.PathModel

**Note:** The constructor of the multiRouter.bicycle.PathModel class is hidden, as this class is not intended for autonomous initialization.

Extends [IEventEmitter](#).

Data model for the path of a bicycle route. A single route can contain several paths, and each path connects two waypoints.

#### Fields

Creates the data model of a bicycle route path.

#### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

### multiRouter.bicycle.Route

**Note:** The constructor of the multiRouter.bicycle.Route class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeoObject](#).

Representation of an individual bicycle route. A multi-stop route can consist of several individual routes.

#### [Fields](#) | [Events](#) | [Methods](#)

Creates a representation to display a single bicycle route.

#### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">IDataManager</a>	Geo object data. Inherited from <a href="#">IGeoObject</a> .
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

## Events

Name	Description
<a href="#">balloonclose</a>	Closing the balloon.
<a href="#">balloonopen</a>	Opening the balloon.
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">update</a>	<p>Updating the route rendering. Instance of the <a href="#">Event</a> class.</p>

Name	Description
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	<p>Returns reference to the map.</p> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	<p>Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	<p>The method provides synchronous access to the overlay.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	<p>Returns link to the parent object, or null if the parent element was not set.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	<p>Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>

## Events details

### balloonclose

Closing the balloon.

### balloonopen

Opening the balloon.

### update

Updating the route rendering. Instance of the [Event](#) class.

### multiRouter.bicycle.RouteModel

**Note:** The constructor of the multiRouter.bicycle.RouteModel class is hidden, as this class is not intended for autonomous initialization.

Extends [IEventEmitter](#).

Data model for an individual bicycle route. A multi-stop route can consist of several individual routes.

## Fields

Creates the data model of an individual bicycle route.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

### multiRouter.bicycle.Segment

**Note:** The constructor of the multiRouter.bicycle.Segment class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeoObject](#).

Representation of a segment on the bicycle route. A segment of a bicycle route is a part of the path from one manoeuver to another.

[Fields](#) | [Events](#) | [Methods](#)

Creates a representation to display a segment on the bicycle route.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">IDataManager</a>	Geo object data. Inherited from <a href="#">IGeoObject</a> .
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

### Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li></ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>oldMap - Old map.</li><li>newMap - New map.</li></ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">update</a>	<p>Updating the segment rendering. Instance of the <code>Event</code> class.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IChildOnMap</a> .

**Events details****update**

Updating the segment rendering. Instance of the [Event](#) class.

**multiRouter.bicycle.SegmentModel**

**Note:** The constructor of the multiRouter.bicycle.SegmentModel class is hidden, as this class is not intended for autonomous initialization.

Extends [IEventEmitter](#).

Data model for a segment on the path of a bicycle route. A segment of a bicycle route is a part of the path from one manoeuvre to another.

**Fields**

Creates the data model for a segment on the path of a bicycle route.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

## multiRouter.driving

### multiRouter.driving.Path

**Note:** The constructor of the `multiRouter.driving.Path` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeoObject](#).

Representation of a path on a multi-stop driving route. A single route can contain several paths, and each path connects two waypoints.

[Fields](#) | [Events](#) | [Methods](#)

Creates a representation to display a path of the multi-stop driving route.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">model</a>	<a href="#">multiRouter.driving.PathModel</a>	Data model for the multi-stop route's path.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">data.Manager</a>	Multi-stop route's path data. The following fields are available: <ul style="list-style-type: none"> <li><code>index</code>: Integer - The sequential number of the path in the multi-stop route's corresponding route.</li> <li><code>type</code>: String - Route type identifier, which takes the value "driving" for automobile routes.</li> <li><code>distance</code>: Object - An object with the "text" and "value" fields that describes the length of the path in meters.</li> <li><code>duration</code>: Object - An object with the "text" and "value" fields that describes the travel time of the path in seconds.</li> <li><code>durationInTraffic</code>: Object - An object with the "text" and "value" fields that specifies the travel time of the path (in seconds) considering traffic.</li> <li><code>coordinates</code>: Number[][] - Coordinates of all points on the path.</li> <li><code>encodedCoordinates</code>: String - A string of base64-encoded coordinates for all points on the path.</li> </ul>
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

### Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li></ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>oldMap - Old map.</li><li>newMap - New map.</li></ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">update</a>	<p>Updating the path rendering. Instance of the <code>Event</code> class.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">getSegments()</a>	<a href="#">GeoObjectCollection</a>	Returns the child collection of segments that the path consists of.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IChildOnMap</a> .

**Fields details****model**

```
{multiRouter.driving.PathModel} model
```

Data model for the multi-stop route's path.

**properties**

```
{data.Manager} properties
```

Multi-stop route's path data. The following fields are available:

- **index:** Integer - The sequential number of the path in the multi-stop route's corresponding route.
- **type:** String - Route type identifier, which takes the value "driving" for automobile routes.
- **distance:** Object - An object with the "text" and "value" fields that describes the length of the path in meters.
- **duration:** Object - An object with the "text" and "value" fields that describes the travel time of the path in seconds.
- **durationInTraffic:** Object - An object with the "text" and "value" fields that specifies the travel time of the path (in seconds) considering traffic.
- **coordinates:** Number[][] - Coordinates of all points on the path.
- **encodedCoordinates:** String - A string of base64-encoded coordinates for all points on the path.

## Events details

### update

Updating the path rendering. Instance of the [Event](#) class.

## Methods details

### getSegments

```
{GeoObjectCollection} getSegments()
```

**Returns** the child collection of segments that the path consists of.

### multiRouter.driving.PathModel

**Note:** The constructor of the `multiRouter.driving.PathModel` class is hidden, as this class is not intended for autonomous initialization.

Extends [IEventEmitter](#).

Data model for the path of a driving route. A single route can contain several paths, and each path connects two waypoints.

[Fields](#) | [Events](#) | [Methods](#)

Creates the data model of a driving route path.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">properties</a>	<a href="#">data.Manager</a>	Multi-stop route's path data. The following fields are available: <ul style="list-style-type: none"> <li><code>index</code>: Integer - The sequential number of the path in the multi-stop route's corresponding route.</li> <li><code>type</code>: String - Route type identifier, which takes the value "driving" for automobile routes.</li> <li><code>distance</code>: Object - An object with the "text" and "value" fields that describes the length of the path in meters.</li> <li><code>duration</code>: Object - An object with the "text" and "value" fields that describes the travel time of the path in seconds.</li> <li><code>durationInTraffic</code>: Object - An object with the "text" and "value" fields that specifies the travel time of the path (in seconds) considering traffic.</li> <li><code>coordinates</code>: Number[][] - Coordinates of all points on the path.</li> <li><code>encodedCoordinates</code>: String - A string of base64-encoded coordinates for all points on the path.</li> </ul>
<a href="#">route</a>	<a href="#">multiRouter.driving.RouteModel</a>	Reference to the parent model of the route.

## Events

Name	Description
<a href="#">update</a>	Updating the model with new data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>segmentsChange</code>: Boolean - Flag for whether the set of segments is changed</li> </ul>

**Methods**

Name	Returns	Description
<a href="#">destroy()</a>		Destroys a model.
<a href="#">getSegments()</a>	<a href="#">multiRouter.driving.SegmentModel[]</a>	Returns array of path segments.
<a href="#">getType()</a>	String	Returns ID of the route path type. For automobile routes, it returns the string "driving".
<a href="#">update(pathJson)</a>		Updates the state of the model.

**Fields details****properties**

```
{data.Manager} properties
```

Multi-stop route's path data. The following fields are available:

- **index**: Integer - The sequential number of the path in the multi-stop route's corresponding route.
- **type**: String - Route type identifier, which takes the value "driving" for automobile routes.
- **distance**: Object - An object with the "text" and "value" fields that describes the length of the path in meters.
- **duration**: Object - An object with the "text" and "value" fields that describes the travel time of the path in seconds.
- **durationInTraffic**: Object - An object with the "text" and "value" fields that specifies the travel time of the path (in seconds) considering traffic.
- **coordinates**: Number[][] - Coordinates of all points on the path.
- **encodedCoordinates**: String - A string of base64-encoded coordinates for all points on the path.

**route**

```
{multiRouter.driving.RouteModel} route
```

Reference to the parent model of the route.

**Events details****update**

Updating the model with new data. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- **segmentsChange**: Boolean - Flag for whether the set of segments is changed

**Methods details****destroy**

```
{ } destroy()
```

Destroys a model.

**getSegments**

```
{multiRouter.driving.SegmentModel[]} getSegments()
```

**Returns** array of path segments.

**getType**

```
{String} getType()
```

**Returns** ID of the route path type. For automobile routes, it returns the string "driving".

**update**

```
{ } update (pathJson)
```

Updates the state of the model.

**Parameters:**

Parameter	Default value	Description
<a href="#">pathJson</a> *	—	Type: Object JSON data.

\* Mandatory parameter/option.

**multiRouter.driving.Route**

**Note:** The constructor of the `multiRouter.driving.Route` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeoObject](#).

Representation of an individual automobile route. A multi-stop route can consist of several individual routes.

[Fields](#) | [Events](#) | [Methods](#)

Creates a representation to display a single driving route.

**Fields**

Name	Type	Description
<a href="#">balloon</a>	<a href="#">IMultiRouterRouteBalloon</a>	Balloon of a route.
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">model</a>	<a href="#">multiRouter.driving.RouteModel</a>	The data model of an individual route.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

Name	Type	Description
<a href="#">properties</a>	<a href="#">data.Manager</a>	<p>The route data. The following fields are available:</p> <ul style="list-style-type: none"> <li>index: Integer - The ordinal number of the route in a multi-stop route.</li> <li>type: String - Route type identifier, which takes the value "driving" for automobile routes.</li> <li>blocked: Boolean - Indicates that the route contains blocked sections.</li> <li>distance: Object - An object with the "text" and "value" fields that specifies the length of the route in meters.</li> <li>duration: Object - An object with the "text" and "value" fields that specifies the travel time of the route in seconds.</li> <li>durationInTraffic: Object - An object with the "text" and "value" fields that specifies the travel time of the route (in seconds) considering traffic.</li> <li>boundedBy: Number[][] - Coordinates of the upper and lower corners of the rectangle that bounds the route.</li> </ul>
<a href="#">state</a>	<a href="#">IDataManager</a>	<p>State of the geo object.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>

## Events

Name	Description
<a href="#">balloonclose</a>	Closing the balloon.
<a href="#">balloonopen</a>	Opening the balloon.
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>

Name	Description
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">update</a>	<p>Updating the route rendering. Instance of the <a href="#">Event</a> class.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	<p>Returns reference to the map.</p> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	<p>Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	<p>The method provides synchronous access to the overlay.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	<p>Returns link to the parent object, or null if the parent element was not set.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>
<a href="#">getPaths()</a>	<a href="#">GeoObjectCollection</a>	<p>Returns a child collection of paths that make up the route.</p>

Name	Returns	Description
<code>setParent(parent)</code>	<code>IChildOnMap</code>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <code>IChildOnMap</code> .

## Fields details

### balloon

```
{IMultiRouterRouteBalloon} balloon
```

Balloon of a route.

### model

```
{multiRouter.driving.RouteModel} model
```

The data model of an individual route.

## properties

```
{data.Manager} properties
```

The route data. The following fields are available:

- `index`: Integer - The ordinal number of the route in a multi-stop route.
- `type`: String - Route type identifier, which takes the value "driving" for automobile routes.
- `blocked`: Boolean - Indicates that the route contains blocked sections.
- `distance`: Object - An object with the "text" and "value" fields that specifies the length of the route in meters.
- `duration`: Object - An object with the "text" and "value" fields that specifies the travel time of the route in seconds.
- `durationInTraffic`: Object - An object with the "text" and "value" fields that specifies the travel time of the route (in seconds) considering traffic.
- `boundedBy`: Number[][] - Coordinates of the upper and lower corners of the rectangle that bounds the route.

## Events details

### balloonclose

Closing the balloon.

### balloonopen

Opening the balloon.

### update

Updating the route rendering. Instance of the [Event](#) class.

## Methods details

### getPaths

```
{GeoObjectCollection} getPaths()
```

**Returns** a child collection of paths that make up the route.

**multiRouter.driving.RouteModel**

**Note:** The constructor of the `multiRouter.driving.RouteModel` class is hidden, as this class is not intended for autonomous initialization.

Extends [IEventEmitter](#).

Data model for an individual driving route. A multi-stop route can consist of several individual routes.

[Fields](#) | [Events](#) | [Methods](#)

Creates the data model of an individual driving route.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">multiRoute</a>	<a href="#">multiRouter.MultiRouteModel</a>	Reference to the parent model of a multi-stop route.
<a href="#">properties</a>	<a href="#">data.Manager</a>	The route data. The following fields are available: <ul style="list-style-type: none"> <li><code>index</code>: Integer - The ordinal number of the route in a multi-stop route.</li> <li><code>type</code>: String - Route type identifier, which takes the value "driving" for automobile routes.</li> <li><code>blocked</code>: Boolean - Indicates that the route contains blocked sections.</li> <li><code>distance</code>: Object - An object with the "text" and "value" fields that specifies the length of the route in meters.</li> <li><code>duration</code>: Object - An object with the "text" and "value" fields that specifies the travel time of the route in seconds.</li> <li><code>durationInTraffic</code>: Object - An object with the "text" and "value" fields that specifies the travel time of the route (in seconds) considering traffic.</li> <li><code>boundedBy</code>: Number[][] - Coordinates of the upper and lower corners of the rectangle that bounds the route.</li> </ul>

**Events**

Name	Description
<a href="#">update</a>	Updating the model with new data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>pathsChange</code>: Boolean - Flag for whether the set of paths is changed.</li> </ul>

**Methods**

Name	Returns	Description
<a href="#">destroy()</a>		Destroys a model.
<a href="#">getPaths()</a>	<a href="#">multiRouter.driving.PathModel</a> []	Returns array of route paths.
<a href="#">getType()</a>	String	Returns ID of the route type. For automobile routes, it returns the string "driving".
<a href="#">update(routeJson)</a>		Updates the state of the model.

## Fields details

### multiRoute

```
{multiRouter.MultiRouteModel} multiRoute
```

Reference to the parent model of a multi-stop route.

### properties

```
{data.Manager} properties
```

The route data. The following fields are available:

- **index**: Integer - The ordinal number of the route in a multi-stop route.
- **type**: String - Route type identifier, which takes the value "driving" for automobile routes.
- **blocked**: Boolean - Indicates that the route contains blocked sections.
- **distance**: Object - An object with the "text" and "value" fields that specifies the length of the route in meters.
- **duration**: Object - An object with the "text" and "value" fields that specifies the travel time of the route in seconds.
- **durationInTraffic**: Object - An object with the "text" and "value" fields that specifies the travel time of the route (in seconds) considering traffic.
- **boundedBy**: Number[][] - Coordinates of the upper and lower corners of the rectangle that bounds the route.

## Events details

### update

Updating the model with new data. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- **pathsChange**: Boolean - Flag for whether the set of paths is changed.

## Methods details

### destroy

```
{ } destroy()
```

Destroys a model.

### getPaths

```
{multiRouter.driving.PathModel[]} getPaths()
```

Returns array of route paths.

### getType

```
{String} getType()
```

Returns ID of the route type. For automobile routes, it returns the string "driving".

### update

```
{ } update(routeJson)
```

Updates the state of the model.

### Parameters:

Parameter	Default value	Description
<code>routeJson</code> *	—	Type: Object JSON data.

\* Mandatory parameter/option.

### **multiRouter.driving.Segment**

**Note:** The constructor of the `multiRouter.driving.Segment` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeoObject](#).

Representation of a segment on the automobile route. A segment of a driving route is a part of the path from one manoeuver to another.

[Fields](#) | [Events](#) | [Methods](#)

Creates a representation to display a segment on the driving route.

### **Fields**

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<code>geometry</code>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<code>model</code>	<a href="#">multiRouter.driving.SegmentModel</a>	Data model for a segment.
<code>options</code>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<code>properties</code>	<a href="#">data.Manager</a>	Segment data. The following fields are available: <ul style="list-style-type: none"> <li><code>index</code>: Integer - The ordinal number of the segment in the array of segments of the corresponding route path.</li> <li><code>type</code>: String - Segment type identifier, which takes the value "driving" for automobile segments.</li> <li><code>street</code>: String - Text description of the street that the segment goes along.</li> <li><code>action</code>: Object - An object with the "text" and "value" fields that describes the final maneuver of the segment.</li> <li><code>distance</code>: Object - An object with the "text" and "value" fields that specifies the length of the segment in meters.</li> <li><code>duration</code>: Object - An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.</li> <li><code>durationInTraffic</code>: Object - An object with the "text" and "value" fields that specifies the travel time of the segment (in seconds) considering traffic.</li> <li><code>text</code>: String - Text description of the segment.</li> <li><code>viaPoints</code>: Integer[] - Indexes of throughpoints lying on the segment.</li> <li><code>lodIndex</code>: Integer - The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.</li> </ul>
<code>state</code>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

## Events

Name	Description
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">update</a>	<p>Updating the segment rendering. Instance of the <code>Event</code> class.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IChildOnMap</a> .

**Fields details****model**

```
{multiRouter.driving.SegmentModel} model
```

Data model for a segment.

**properties**

```
{data.Manager} properties
```

Segment data. The following fields are available:

- **index:** Integer - The ordinal number of the segment in the array of segments of the corresponding route path.
- **type:** String - Segment type identifier, which takes the value "driving" for automobile segments.
- **street:** String - Text description of the street that the segment goes along.
- **action:** Object - An object with the "text" and "value" fields that describes the final maneuver of the segment.
- **distance:** Object - An object with the "text" and "value" fields that specifies the length of the segment in meters.
- **duration:** Object - An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.
- **durationInTraffic:** Object - An object with the "text" and "value" fields that specifies the travel time of the segment (in seconds) considering traffic.
- **text:** String - Text description of the segment.
- **viaPoints:** Integer[] - Indexes of throughpoints lying on the segment.

- `lodIndex`: Integer - The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

### Events details

#### update

Updating the segment rendering. Instance of the [Event](#) class.

### multiRouter.driving.SegmentModel

**Note:** The constructor of the `multiRouter.driving.SegmentModel` class is hidden, as this class is not intended for autonomous initialization.

Extends [IEventEmitter](#).

Data model for a segment on the path of a driving route. A segment of a driving route is a part of the path from one manoeuver to another.

[Fields](#) | [Events](#) | [Methods](#)

Creates the data model for a segment on the path of a driving route.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">geometry.base.LineString</a>	Geometry of a segment.
<a href="#">path</a>	<a href="#">multiRouter.driving.PathModel</a>	Reference to the parent model of the path.
<a href="#">properties</a>	<a href="#">data.Manager</a>	Segment data. The following fields are available: <ul style="list-style-type: none"> <li>• <code>index</code>: Integer - The ordinal number of the segment in the array of segments of the corresponding route path.</li> <li>• <code>type</code>: String - Segment type identifier, which takes the value "driving" for automobile segments.</li> <li>• <code>street</code>: String - Text description of the street that the segment goes along.</li> <li>• <code>action</code>: Object - An object with the "text" and "value" fields that describes the final maneuver of the segment.</li> <li>• <code>distance</code>: Object - An object with the "text" and "value" fields that specifies the length of the segment in meters.</li> <li>• <code>duration</code>: Object - An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.</li> <li>• <code>durationInTraffic</code>: Object - An object with the "text" and "value" fields that specifies the travel time of the segment (in seconds) considering traffic.</li> <li>• <code>text</code>: String - Text description of the segment.</li> <li>• <code>viaPoints</code>: Integer[] - Indexes of throughpoints lying on the segment.</li> <li>• <code>lodIndex</code>: Integer - The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.</li> </ul>

### Events

Name	Description
<a href="#">update</a>	Updating the model with new data. Instance of the <a href="#">Event</a> class.

**Methods**

Name	Returns	Description
<a href="#">destroy()</a>		Destroys a model.
<a href="#">getType()</a>	String	Returns ID of the segment type. For segments of automobile routes, it returns the string "driving".
<a href="#">getViaPoints()</a>	<a href="#">multiRouter.ViaPointModel[]</a>	Returns an array of throughpoints on the segment.
<a href="#">update(segmentJson)</a>		Updates the state of the model.

**Fields details****geometry**

```
{geometry.base.LineString} geometry
```

Geometry of a segment.

**path**

```
{multiRouter.driving.PathModel} path
```

Reference to the parent model of the path.

**properties**

```
{data.Manager} properties
```

Segment data. The following fields are available:

- **index**: Integer - The ordinal number of the segment in the array of segments of the corresponding route path.
- **type**: String - Segment type identifier, which takes the value "driving" for automobile segments.
- **street**: String - Text description of the street that the segment goes along.
- **action**: Object - An object with the "text" and "value" fields that describes the final maneuver of the segment.
- **distance**: Object - An object with the "text" and "value" fields that specifies the length of the segment in meters.
- **duration**: Object - An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.
- **durationInTraffic**: Object - An object with the "text" and "value" fields that specifies the travel time of the segment (in seconds) considering traffic.
- **text**: String - Text description of the segment.
- **viaPoints**: Integer[] - Indexes of throughpoints lying on the segment.
- **lodIndex**: Integer - The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

**Events details****update**

Updating the model with new data. Instance of the [Event](#) class.

## Methods details

### destroy

```
{ } destroy()
```

Destroys a model.

### getType

```
{String} getType()
```

**Returns** ID of the segment type. For segments of automobile routes, it returns the string "driving".

### getViaPoints

```
{multiRouter.ViaPointModel[]} getViaPoints()
```

**Returns** an array of throughpoints on the segment.

### update

```
{ } update(segmentJson)
```

Updates the state of the model.

#### Parameters:

Parameter	Default value	Description
<a href="#">segmentJson</a> *	—	Type: Object JSON data.

\* Mandatory parameter/option.

## multiRouter.Editor

Extends [ICustomizable](#), [IEventEmitter](#).

Multi-route editor.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
multiRouter.Editor(multiRoute[, state[, options]])
```

Creates a multi-route editor.

#### Parameters:

Parameter	Default value	Description
<a href="#">multiRoute</a> *	—	Type: <a href="#">multiRouter.MultiRoute</a> The multi-route being edited.
<a href="#">state</a>	—	Type: Object Object which describes the initial state of the editor. For a list of available fields, see <a href="#">multiRouter.Editor.state</a> .

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.drawCursor</a>	—	Type: Object Name of the cursor to use in waypoint drawing mode.
<a href="#">options.drawOver</a>	true	Type: Object Allows putting points on top of map objects in waypoint drawing mode.
<a href="#">options.midPointsType</a>	"way"	Type: String Specifies the type of points to add while dragging the marker which appears when hovering the mouse cursor over the active route. Accepts one of the following string values: <ul style="list-style-type: none"> <li>"way" - Add waypoints.</li> <li>"via" - Add throughpoints.</li> </ul> See also the description of the <code>addMidPoints</code> field for the state manager <a href="#">multiRouter.Editor.state</a> .

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">state</a>	<a href="#">data.Manager</a>	The state manager of the multi-stop route editor. Available fields: <ul style="list-style-type: none"> <li><code>addWayPoints</code>: Boolean - Allows adding new waypoints by clicking on the map. Default value: false.</li> <li><code>dragWayPoints</code>: Boolean - Allows dragging existing waypoints. Default value: true.</li> <li><code>removeWayPoints</code>: Boolean - Allows deleting waypoints by double-clicking on them. Default value: false.</li> <li><code>dragViaPoints</code>: Boolean - Allows dragging existing throughpoints. Default value: true.</li> <li><code>removeViaPoints</code>: Boolean - Allows deleting throughpoints by double-clicking on them. Default value: true.</li> <li><code>addMidPoints</code>: Boolean - Allows adding intermediate throughpoints or waypoints by dragging the placemark which appears when you hover the mouse over the active route. The type of point to add is set by the option <code>midPointsType</code>. Default value: true.</li> </ul>

## Events

Name	Description
<a href="#">beforemidpointadd</a>	<p>Event preceding the "midpointadd" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>coords</code>: <code>Number[]</code> - Coordinates for adding a midpoint.</li> <li>• <code>pointType</code>: <code>String</code> - Type ID for the midpoint being added.</li> <li>• <code>insertIndex</code>: <code>Integer</code> - The insertion index of the midpoint in the set of reference points for the multi-stop route.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>• <code>setPointType</code> - The method that allows you to set the type of the added point. Takes a string identifier of the type as the argument (see the description of the <code>midPointsType</code> option).</li> <li>• <code>setInsertIndex</code> - The method that allows you to adjust the insertion index of a midpoint before it is applied. Takes the new index as the argument.</li> </ul> <p>If you call the <a href="#">Event.preventDefault</a> method of this event, the next midpoint to add, as well as the "midpointadd" event, will be canceled.</p>
<a href="#">beforemidpointdrag</a>	<p>The event preceding the "midpointdrag" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>pixelOffset</code>: <code>Number[]</code> - Pixel shift along the axes at the given step.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>• <code>setPixelOffset</code> - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, the subsequent "midpointdrag" event will be canceled.</p>
<a href="#">beforemidpointpinshow</a>	<p>The event preceding the "midpointpinshow" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>pin</code>: <a href="#">Placemark</a> - Reference to the placemark object.</li> <li>• <code>globalPixels</code>: <code>Number[]</code> - Global pixel coordinates of the placemark.</li> <li>• <code>segment</code>: <a href="#">multiRouter.driving.Segment</a> - Reference to the segment of the route where the placemark should appear.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent "midpointpinshow" event will be canceled and the placemark will be hidden.</p>
<a href="#">beforeviapointdrag</a>	<p>The event preceding the "viapointdrag" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>pixelOffset</code>: <code>Number[]</code> - Pixel shift along the axes at the given step.</li> <li>• <code>viaPoint</code>: <a href="#">multiRouter.ViaPoint</a> - A reference to the throughpoint object being dragged.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>• <code>setPixelOffset</code> - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent "viapointdrag" event will be canceled.</p>

Name	Description
<a href="#">beforeviapointdragstart</a>	<p>The event preceding the "viapointdragstart" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• domEvent: <a href="#">DomEvent</a> - The source DOM event, if there is one.</li> <li>• viaPoint: <a href="#">multiRouter.ViaPoint</a> - A reference to the throughpoint object being dragged.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, any subsequent dragging, as well as the "viapointdragstart" event, will be canceled.</p>
<a href="#">beforeviapointremove</a>	<p>The event preceding the "viapointremove" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• viaPoint: <a href="#">multiRouter.ViaPoint</a> - A reference to the throughpoint object being deleted.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, deletion of a throughpoint, as well as a subsequent "viapointremove" event, will be canceled.</p>
<a href="#">beforewaypointadd</a>	<p>Event preceding the "waypointadd" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• coords: <code>Number[]</code> - Coordinates of the added waypoint.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>• setCoords - Method that allows you to adjust the coordinates of the waypoint being added. It takes an argument with the new pixel shift in the form of an array of two numbers.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, addition of a waypoint, as well as a subsequent "waypointadd" event, will be canceled.</p>
<a href="#">beforewaypointdrag</a>	<p>The event preceding the "waypointdrag" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• pixelOffset: <code>Number[]</code> - Pixel shift along the axes at the given step.</li> <li>• wayPoint: <a href="#">multiRouter.WayPoint</a> - A reference to the waypoint object being dragged.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>• setPixelOffset - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent "waypointdrag" event will be canceled.</p>
<a href="#">beforewaypointdragstart</a>	<p>The event preceding the "waypointdragstart" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• domEvent: <a href="#">DomEvent</a> - The source DOM event, if there is one.</li> <li>• wayPoint: <a href="#">multiRouter.WayPoint</a> - A reference to the waypoint object being dragged.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, any subsequent dragging, as well as the "waypointdragstart" event, will be canceled.</p>
<a href="#">beforewaypointremove</a>	<p>The event preceding the "waypointremove" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• wayPoint: <a href="#">multiRouter.WayPoint</a> - A reference to the waypoint object being deleted.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, removal of the waypoint, as well as a subsequent "waypointremove" event, will be canceled.</p>

Name	Description
<a href="#">midpointadd</a>	<p>Adding a midpoint (intermediate point) to the route. The point type is determined by the value of the <code>midPointsType</code> option. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>coords</code>: <code>Number[]</code> - Coordinates for adding a midpoint.</li> <li><code>pointType</code>: <code>String</code> - Type ID for the midpoint being added.</li> <li><code>insertIndex</code>: <code>Integer</code> - The insertion index of the midpoint in the set of reference points for the multi-stop route.</li> </ul>
<a href="#">midpointdrag</a>	<p>Dragging the added midpoint. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>pixelOffset</code>: <code>Number[]</code> - Pixel shift along the axes at the given step.</li> </ul>
<a href="#">midpointdragend</a>	<p>End of dragging the added midpoint. Instance of the <code>Event</code> class.</p>
<a href="#">midpointpinshow</a>	<p>Displaying the draggable placemark when you hover over the active route. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>pin</code>: <code>Placemark</code> - Reference to the placemark object.</li> <li><code>globalPixels</code>: <code>Number[]</code> - Global pixel coordinates of the placemark.</li> <li><code>segment</code>: <code>multiRouter.driving.Segment</code> - Reference to the segment of the route where the placemark appeared.</li> </ul>
<a href="#">optionschange</a>	<p>Change to the object options. Inherited from <code>ICustomizable</code>.</p>
<a href="#">viapointdrag</a>	<p>Throughpoint dragging. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>pixelOffset</code>: <code>Number[]</code> - Pixel shift along the axes at the given step.</li> <li><code>viaPoint</code>: <code>multiRouter.ViaPoint</code> - A reference to the throughpoint object being dragged.</li> </ul>
<a href="#">viapointdragend</a>	<p>End of throughpoint dragging. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>viaPoint</code>: <code>multiRouter.ViaPoint</code> - A reference to the throughpoint object being dragged.</li> </ul>
<a href="#">viapointdragstart</a>	<p>Start of throughpoint dragging. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>domEvent</code>: <code>DomEvent</code> - The source DOM event, if there is one.</li> <li><code>viaPoint</code>: <code>multiRouter.ViaPoint</code> - A reference to the throughpoint object being dragged.</li> </ul>
<a href="#">viapointremove</a>	<p>Deleting a throughpoint. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>viaPoint</code>: <code>multiRouter.ViaPoint</code> - A reference to the deleted throughpoint object.</li> </ul>
<a href="#">waypointadd</a>	<p>Adding a waypoint. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>coords</code>: <code>Number[]</code> - Coordinates of the added waypoint.</li> </ul>
<a href="#">waypointdrag</a>	<p>Waypoint dragging. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>pixelOffset</code>: <code>Number[]</code> - Pixel shift along the axes at the given step.</li> <li><code>wayPoint</code>: <code>multiRouter.WayPoint</code> - A reference to the waypoint object being dragged.</li> </ul>

Name	Description
<a href="#">waypointdragend</a>	End of waypoint dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>wayPoint: <a href="#">multiRouter.WayPoint</a> - A reference to the waypoint object being dragged.</li> </ul>
<a href="#">waypointdragstart</a>	Start of waypoint dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>domEvent: <a href="#">DomEvent</a> - The source DOM event, if there is one.</li> <li>wayPoint: <a href="#">multiRouter.WayPoint</a> - A reference to the waypoint object being dragged.</li> </ul>
<a href="#">waypointremove</a>	Deleting a waypoint. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>wayPoint: <a href="#">multiRouter.WayPoint</a> - A reference to the deleted waypoint object.</li> </ul>

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destroys the multi-route editor.
<a href="#">getMultiRoute()</a>	<a href="#">multiRouter.MultiRoute</a>	Returns a reference to the multi-route being edited.

## Fields details

### state

```
{data.Manager} state
```

The state manager of the multi-stop route editor.

Available fields:

- addWayPoints: Boolean - Allows adding new waypoints by clicking on the map. Default value: false.
- dragWayPoints: Boolean - Allows dragging existing waypoints. Default value: true.
- removeWayPoints: Boolean - Allows deleting waypoints by double-clicking on them. Default value: false.
- dragViaPoints: Boolean - Allows dragging existing throughpoints. Default value: true.
- removeViaPoints: Boolean - Allows deleting throughpoints by double-clicking on them. Default value: true.
- addMidPoints: Boolean - Allows adding intermediate throughpoints or waypoints by dragging the placemark which appears when you hover the mouse over the active route. The type of point to add is set by the option `midPointsType`. Default value: true.

## Events details

### beforemidpointadd

Event preceding the "midpointadd" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- coords: Number[] - Coordinates for adding a midpoint.
- pointType: String - Type ID for the midpoint being added.
- insertIndex: Integer - The insertion index of the midpoint in the set of reference points for the multi-stop route.

Names of methods that are accessible via [Event.callMethod](#):

- `setPointType` - The method that allows you to set the type of the added point. Takes a string identifier of the type as the argument (see the description of the `midPointsType` option).
- `setInsertIndex` - The method that allows you to adjust the insertion index of a midpoint before it is applied. Takes the new index as the argument.

If you call the `Event.preventDefault` method of this event, the next midpoint to add, as well as the "midpointadd" event, will be canceled.

#### **beforemidpointdrag**

The event preceding the "midpointdrag" event. Instance of the `Event` class. Names of fields that are available via the `Event.get` method:

- `pixelOffset`: `Number[]` - Pixel shift along the axes at the given step.

Names of methods that are accessible via `Event.callMethod`:

- `setPixelOffset` - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the `Event.preventDefault` method is called for this event, the subsequent "midpointdrag" event will be canceled.

#### **beforemidpointpinshow**

The event preceding the "midpointpinshow" event. Instance of the `Event` class. Names of fields that are available via the `Event.get` method:

- `pin`: `Placemark` - Reference to the placemark object.
- `globalPixels`: `Number[]` - Global pixel coordinates of the placemark.
- `segment`: `multiRouter.driving.Segment` - Reference to the segment of the route where the placemark should appear.

If the `Event.preventDefault` method is called for this event, a subsequent "midpointpinshow" event will be canceled and the placemark will be hidden.

#### **beforeviapointdrag**

The event preceding the "viapointdrag" event. Instance of the `Event` class. Names of fields that are available via the `Event.get` method:

- `pixelOffset`: `Number[]` - Pixel shift along the axes at the given step.
- `viaPoint`: `multiRouter.ViaPoint` - A reference to the throughpoint object being dragged.

Names of methods that are accessible via `Event.callMethod`:

- `setPixelOffset` - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the `Event.preventDefault` method is called for this event, a subsequent "viapointdrag" event will be canceled.

#### **beforeviapointdragstart**

The event preceding the "viapointdragstart" event. Instance of the `Event` class. Names of fields that are available via the `Event.get` method:

- `domEvent`: `DomEvent` - The source DOM event, if there is one.
- `viaPoint`: `multiRouter.ViaPoint` - A reference to the throughpoint object being dragged.

If the `Event.preventDefault` method is called for this event, any subsequent dragging, as well as the "viapointdragstart" event, will be canceled.

#### **beforeviapointremove**

The event preceding the "viapointremove" event. Instance of the `Event` class. Names of fields that are available via the `Event.get` method:

- `viaPoint`: [multiRouter.ViaPoint](#) - A reference to the throughpoint object being deleted.

If the [Event.preventDefault](#) method is called for this event, deletion of a throughpoint, as well as a subsequent "viapointremove" event, will be canceled.

### **beforewaypointadd**

Event preceding the "waypointadd" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `coords`: `Number[]` - Coordinates of the added waypoint.

Names of methods that are accessible via [Event.callMethod](#):

- `setCoords` - Method that allows you to adjust the coordinates of the waypoint being added. It takes an argument with the new pixel shift in the form of an array of two numbers.

If the [Event.preventDefault](#) method is called for this event, addition of a waypoint, as well as a subsequent "waypointadd" event, will be canceled.

### **beforewaypointdrag**

The event preceding the "waypointdrag" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `pixelOffset`: `Number[]` - Pixel shift along the axes at the given step.
- `wayPoint`: [multiRouter.WayPoint](#) - A reference to the waypoint object being dragged.

Names of methods that are accessible via [Event.callMethod](#):

- `setPixelOffset` - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the [Event.preventDefault](#) method is called for this event, a subsequent "waypointdrag" event will be canceled.

### **beforewaypointdragstart**

The event preceding the "waypointdragstart" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `domEvent`: [DomEvent](#) - The source DOM event, if there is one.
- `wayPoint`: [multiRouter.WayPoint](#) - A reference to the waypoint object being dragged.

If the [Event.preventDefault](#) method is called for this event, any subsequent dragging, as well as the "waypointdragstart" event, will be canceled.

### **beforewaypointremove**

The event preceding the "waypointremove" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `wayPoint`: [multiRouter.WayPoint](#) - A reference to the waypoint object being deleted.

If the [Event.preventDefault](#) method is called for this event, removal of the waypoint, as well as a subsequent "waypointremove" event, will be canceled.

### **midpointadd**

Adding a midpoint (intermediate point) to the route. The point type is determined by the value of the `midPointsType` option. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `coords`: `Number[]` - Coordinates for adding a midpoint.
- `pointType`: `String` - Type ID for the midpoint being added.
- `insertIndex`: `Integer` - The insertion index of the midpoint in the set of reference points for the multi-stop route.

### midpointdrag

Dragging the added midpoint. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- pixelOffset: [Number\[\]](#) - Pixel shift along the axes at the given step.

### midpointdragend

End of dragging the added midpoint. Instance of the [Event](#) class.

### midpointpinshow

Displaying the draggable placemark when you hover over the active route. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- pin: [Placemark](#) - Reference to the placemark object.
- globalPixels: [Number\[\]](#) - Global pixel coordinates of the placemark.
- segment: [multiRouter.driving.Segment](#) - Reference to the segment of the route where the placemark appeared.

### viapointdrag

Throughpoint dragging. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- pixelOffset: [Number\[\]](#) - Pixel shift along the axes at the given step.
- viaPoint: [multiRouter.ViaPoint](#) - A reference to the throughpoint object being dragged.

### viapointdragend

End of throughpoint dragging. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- viaPoint: [multiRouter.ViaPoint](#) - A reference to the throughpoint object being dragged.

### viapointdragstart

Start of throughpoint dragging. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- domEvent: [DomEvent](#) - The source DOM event, if there is one.
- viaPoint: [multiRouter.ViaPoint](#) - A reference to the throughpoint object being dragged.

### viapointremove

Deleting a throughpoint. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- viaPoint: [multiRouter.ViaPoint](#) - A reference to the deleted throughpoint object.

### waypointadd

Adding a waypoint. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- coords: [Number\[\]](#) - Coordinates of the added waypoint.

### waypointdrag

Waypoint dragging. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- pixelOffset: [Number\[\]](#) - Pixel shift along the axes at the given step.
- wayPoint: [multiRouter.WayPoint](#) - A reference to the waypoint object being dragged.

**waypointdragend**

End of waypoint dragging. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `wayPoint`: [multiRouter.WayPoint](#) - A reference to the waypoint object being dragged.

**waypointdragstart**

Start of waypoint dragging. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `domEvent`: [DomEvent](#) - The source DOM event, if there is one.
- `wayPoint`: [multiRouter.WayPoint](#) - A reference to the waypoint object being dragged.

**waypointremove**

Deleting a waypoint. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `wayPoint`: [multiRouter.WayPoint](#) - A reference to the deleted waypoint object.

**Methods details****destroy**

```
{ } destroy()
```

Destroys the multi-route editor.

**getMultiRoute**

```
{multiRouter.MultiRoute} getMultiRoute()
```

**Returns** a reference to the multi-route being edited.

**multiRouter.EditorAddon**

**Note:** The constructor of the `multiRouter.EditorAddon` class is hidden, as this class is not intended for autonomous initialization.

Extends [ICustomizable](#), [IEventEmitter](#).

Add-on for the multi-stop route editor.

[Fields](#) | [Events](#) | [Methods](#)

Creates an add-on for the multi-stop route editor.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">state</a>	<a href="#">data.Manager</a>	The state manager of the multi-stop route editor.

## Events

Name	Description
<a href="#">beforemidpointadd</a>	<p>Event preceding the "midpointadd" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>coords</code>: <code>Number[]</code> - Coordinates for adding a midpoint.</li> <li>• <code>pointType</code>: <code>String</code> - Type ID for the midpoint being added.</li> <li>• <code>insertIndex</code>: <code>Integer</code> - The insertion index of the midpoint in the set of reference points for the multi-stop route.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>• <code>setPointType</code> - The method that allows you to set the type of the added point. Takes a string identifier of the type as the argument (see the description of the <code>midPointsType</code> option).</li> <li>• <code>setInsertIndex</code> - The method that allows you to adjust the insertion index of a midpoint before it is applied. Takes the new index as the argument.</li> </ul> <p>If you call the <a href="#">Event.preventDefault</a> method of this event, the next midpoint to add, as well as the "midpointadd" event, will be canceled.</p>
<a href="#">beforemidpointdrag</a>	<p>The event preceding the "midpointdrag" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>pixelOffset</code>: <code>Number[]</code> - Pixel shift along the axes at the given step.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>• <code>setPixelOffset</code> - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, the subsequent "midpointdrag" event will be canceled.</p>
<a href="#">beforemidpointpinshow</a>	<p>The event preceding the "midpointpinshow" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>pin</code>: <a href="#">Placemark</a> - Reference to the placemark object.</li> <li>• <code>globalPixels</code>: <code>Number[]</code> - Global pixel coordinates of the placemark.</li> <li>• <code>segment</code>: <a href="#">multiRouter.driving.Segment</a> - Reference to the segment of the route where the placemark should appear.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent "midpointpinshow" event will be canceled and the placemark will be hidden.</p>
<a href="#">beforeviapointdrag</a>	<p>The event preceding the "viapointdrag" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>pixelOffset</code>: <code>Number[]</code> - Pixel shift along the axes at the given step.</li> <li>• <code>viaPoint</code>: <a href="#">multiRouter.ViaPoint</a> - A reference to the throughpoint object being dragged.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>• <code>setPixelOffset</code> - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent "viapointdrag" event will be canceled.</p>

Name	Description
<a href="#">beforeviapointdragstart</a>	<p>The event preceding the "viapointdragstart" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• domEvent: <a href="#">DomEvent</a> - The source DOM event, if there is one.</li> <li>• viaPoint: <a href="#">multiRouter.ViaPoint</a> - A reference to the throughpoint object being dragged.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, any subsequent dragging, as well as the "viapointdragstart" event, will be canceled.</p>
<a href="#">beforeviapointremove</a>	<p>The event preceding the "viapointremove" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• viaPoint: <a href="#">multiRouter.ViaPoint</a> - A reference to the throughpoint object being deleted.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, deletion of a throughpoint, as well as a subsequent "viapointremove" event, will be canceled.</p>
<a href="#">beforewaypointadd</a>	<p>Event preceding the "waypointadd" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• coords: Number[] - Coordinates of the added waypoint.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>• setCoords - Method that allows you to adjust the coordinates of the waypoint being added. It takes an argument with the new pixel shift in the form of an array of two numbers.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, addition of a waypoint, as well as a subsequent "waypointadd" event, will be canceled.</p>
<a href="#">beforewaypointdrag</a>	<p>The event preceding the "waypointdrag" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• pixelOffset: Number[] - Pixel shift along the axes at the given step.</li> <li>• wayPoint: <a href="#">multiRouter.WayPoint</a> - A reference to the waypoint object being dragged.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>• setPixelOffset - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent "waypointdrag" event will be canceled.</p>
<a href="#">beforewaypointdragstart</a>	<p>The event preceding the "waypointdragstart" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• domEvent: <a href="#">DomEvent</a> - The source DOM event, if there is one.</li> <li>• wayPoint: <a href="#">multiRouter.WayPoint</a> - A reference to the waypoint object being dragged.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, any subsequent dragging, as well as the "waypointdragstart" event, will be canceled.</p>
<a href="#">beforewaypointremove</a>	<p>The event preceding the "waypointremove" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• wayPoint: <a href="#">multiRouter.WayPoint</a> - A reference to the waypoint object being deleted.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, removal of the waypoint, as well as a subsequent "waypointremove" event, will be canceled.</p>

Name	Description
<a href="#">midpointadd</a>	<p>Adding a midpoint (intermediate point) to the route. The point type is determined by the value of the <code>midPointsType</code> option. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>coords</code>: <code>Number[]</code> - Coordinates for adding a midpoint.</li> <li><code>pointType</code>: <code>String</code> - Type ID for the midpoint being added.</li> <li><code>insertIndex</code>: <code>Integer</code> - The insertion index of the midpoint in the set of reference points for the multi-stop route.</li> </ul>
<a href="#">midpointdrag</a>	<p>Dragging the added midpoint. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>pixelOffset</code>: <code>Number[]</code> - Pixel shift along the axes at the given step.</li> </ul>
<a href="#">midpointdragend</a>	<p>End of dragging the added midpoint. Instance of the <code>Event</code> class.</p>
<a href="#">midpointpinshow</a>	<p>Displaying the draggable placemark when you hover over the active route. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>pin</code>: <code>Placemark</code> - Reference to the placemark object.</li> <li><code>globalPixels</code>: <code>Number[]</code> - Global pixel coordinates of the placemark.</li> <li><code>segment</code>: <code>multiRouter.driving.Segment</code> - Reference to the segment of the route where the placemark appeared.</li> </ul>
<a href="#">optionschange</a>	<p>Change to the object options. Inherited from <code>ICustomizable</code>.</p>
<a href="#">start</a>	<p>Enabling the editor. Instance of the <code>Event</code> class.</p>
<a href="#">stop</a>	<p>Disabling the editor. Instance of the <code>Event</code> class.</p>
<a href="#">viapointdrag</a>	<p>Throughpoint dragging. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>pixelOffset</code>: <code>Number[]</code> - Pixel shift along the axes at the given step.</li> <li><code>viaPoint</code>: <code>multiRouter.ViaPoint</code> - A reference to the throughpoint object being dragged.</li> </ul>
<a href="#">viapointdragend</a>	<p>End of throughpoint dragging. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>viaPoint</code>: <code>multiRouter.ViaPoint</code> - A reference to the throughpoint object being dragged.</li> </ul>
<a href="#">viapointdragstart</a>	<p>Start of throughpoint dragging. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>domEvent</code>: <code>DomEvent</code> - The source DOM event, if there is one.</li> <li><code>viaPoint</code>: <code>multiRouter.ViaPoint</code> - A reference to the throughpoint object being dragged.</li> </ul>
<a href="#">viapointremove</a>	<p>Deleting a throughpoint. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>viaPoint</code>: <code>multiRouter.ViaPoint</code> - A reference to the deleted throughpoint object.</li> </ul>
<a href="#">waypointadd</a>	<p>Adding a waypoint. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>coords</code>: <code>Number[]</code> - Coordinates of the added waypoint.</li> </ul>

Name	Description
<a href="#">waypointdrag</a>	Waypoint dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>pixelOffset: Number[] - Pixel shift along the axes at the given step.</li> <li>wayPoint: <a href="#">multiRouter.WayPoint</a> - A reference to the waypoint object being dragged.</li> </ul>
<a href="#">waypointdragend</a>	End of waypoint dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>wayPoint: <a href="#">multiRouter.WayPoint</a> - A reference to the waypoint object being dragged.</li> </ul>
<a href="#">waypointdragstart</a>	Start of waypoint dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>domEvent: <a href="#">DomEvent</a> - The source DOM event, if there is one.</li> <li>wayPoint: <a href="#">multiRouter.WayPoint</a> - A reference to the waypoint object being dragged.</li> </ul>
<a href="#">waypointremove</a>	Deleting a waypoint. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>wayPoint: <a href="#">multiRouter.WayPoint</a> - A reference to the deleted waypoint object.</li> </ul>

## Methods

Name	Returns	Description
<a href="#">isActive()</a>	Boolean	Returns a flag for whether the editor is currently enabled.
<a href="#">start(state)</a>		Enables the editor.
<a href="#">stop()</a>		Disables the editor.

## Fields details

### state

```
{data.Manager} state
```

The state manager of the multi-stop route editor.

## Events details

### beforemidpointadd

Event preceding the "midpointadd" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- coords: Number[] - Coordinates for adding a midpoint.
- pointType: String - Type ID for the midpoint being added.
- insertIndex: Integer - The insertion index of the midpoint in the set of reference points for the multi-stop route.

Names of methods that are accessible via [Event.callMethod](#):

- setPointType - The method that allows you to set the type of the added point. Takes a string identifier of the type as the argument (see the description of the midPointsType option).
- setInsertIndex - The method that allows you to adjust the insertion index of a midpoint before it is applied. Takes the new index as the argument.

If you call the [Event.preventDefault](#) method of this event, the next midpoint to add, as well as the "midpointadd" event, will be canceled.

### beforemidpointdrag

The event preceding the "midpointdrag" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- pixelOffset: Number[] - Pixel shift along the axes at the given step.

Names of methods that are accessible via [Event.callMethod](#):

- setPixelOffset - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the [Event.preventDefault](#) method is called for this event, the subsequent "midpointdrag" event will be canceled.

### beforemidpointpinshow

The event preceding the "midpointpinshow" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- pin: [Placemark](#) - Reference to the placemark object.
- globalPixels: Number[] - Global pixel coordinates of the placemark.
- segment: [multiRouter.driving.Segment](#) - Reference to the segment of the route where the placemark should appear.

If the [Event.preventDefault](#) method is called for this event, a subsequent "midpointpinshow" event will be canceled and the placemark will be hidden.

### beforeviapointdrag

The event preceding the "viapointdrag" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- pixelOffset: Number[] - Pixel shift along the axes at the given step.
- viaPoint: [multiRouter.ViaPoint](#) - A reference to the throughpoint object being dragged.

Names of methods that are accessible via [Event.callMethod](#):

- setPixelOffset - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the [Event.preventDefault](#) method is called for this event, a subsequent "viapointdrag" event will be canceled.

### beforeviapointdragstart

The event preceding the "viapointdragstart" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- domEvent: [DomEvent](#) - The source DOM event, if there is one.
- viaPoint: [multiRouter.ViaPoint](#) - A reference to the throughpoint object being dragged.

If the [Event.preventDefault](#) method is called for this event, any subsequent dragging, as well as the "viapointdragstart" event, will be canceled.

### beforeviapointremove

The event preceding the "viapointremove" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- viaPoint: [multiRouter.ViaPoint](#) - A reference to the throughpoint object being deleted.

If the [Event.preventDefault](#) method is called for this event, deletion of a throughpoint, as well as a subsequent "viapointremove" event, will be canceled.

### beforewaypointadd

Event preceding the "waypointadd" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- coords: Number[] - Coordinates of the added waypoint.

Names of methods that are accessible via [Event.callMethod](#):

- setCoords - Method that allows you to adjust the coordinates of the waypoint being added. It takes an argument with the new pixel shift in the form of an array of two numbers.

If the [Event.preventDefault](#) method is called for this event, addition of a waypoint, as well as a subsequent "waypointadd" event, will be canceled.

### beforewaypointdrag

The event preceding the "waypointdrag" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- pixelOffset: Number[] - Pixel shift along the axes at the given step.
- waypoint: [multiRouter.WayPoint](#) - A reference to the waypoint object being dragged.

Names of methods that are accessible via [Event.callMethod](#):

- setPixelOffset - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.

If the [Event.preventDefault](#) method is called for this event, a subsequent "waypointdrag" event will be canceled.

### beforewaypointdragstart

The event preceding the "waypointdragstart" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- domEvent: [DomEvent](#) - The source DOM event, if there is one.
- waypoint: [multiRouter.WayPoint](#) - A reference to the waypoint object being dragged.

If the [Event.preventDefault](#) method is called for this event, any subsequent dragging, as well as the "waypointdragstart" event, will be canceled.

### beforewaypointremove

The event preceding the "waypointremove" event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- waypoint: [multiRouter.WayPoint](#) - A reference to the waypoint object being deleted.

If the [Event.preventDefault](#) method is called for this event, removal of the waypoint, as well as a subsequent "waypointremove" event, will be canceled.

### midpointadd

Adding a midpoint (intermediate point) to the route. The point type is determined by the value of the `midPointsType` option. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- coords: Number[] - Coordinates for adding a midpoint.
- pointType: String - Type ID for the midpoint being added.
- insertIndex: Integer - The insertion index of the midpoint in the set of reference points for the multi-stop route.

### midpointdrag

Dragging the added midpoint. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- pixelOffset: Number[] - Pixel shift along the axes at the given step.

**midpointdragend**

End of dragging the added midpoint. Instance of the [Event](#) class.

**midpointpinshow**

Displaying the draggable placemark when you hover over the active route. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- pin: [Placemark](#) - Reference to the placemark object.
- globalPixels: [Number\[\]](#) - Global pixel coordinates of the placemark.
- segment: [multiRouter.driving.Segment](#) - Reference to the segment of the route where the placemark appeared.

**start**

Enabling the editor. Instance of the [Event](#) class.

**stop**

Disabling the editor. Instance of the [Event](#) class.

**viapointdrag**

Throughpoint dragging. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- pixelOffset: [Number\[\]](#) - Pixel shift along the axes at the given step.
- viaPoint: [multiRouter.ViaPoint](#) - A reference to the throughpoint object being dragged.

**viapointdragend**

End of throughpoint dragging. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- viaPoint: [multiRouter.ViaPoint](#) - A reference to the throughpoint object being dragged.

**viapointdragstart**

Start of throughpoint dragging. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- domEvent: [DomEvent](#) - The source DOM event, if there is one.
- viaPoint: [multiRouter.ViaPoint](#) - A reference to the throughpoint object being dragged.

**viapointremove**

Deleting a throughpoint. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- viaPoint: [multiRouter.ViaPoint](#) - A reference to the deleted throughpoint object.

**waypointadd**

Adding a waypoint. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- coords: [Number\[\]](#) - Coordinates of the added waypoint.

**waypointdrag**

Waypoint dragging. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- pixelOffset: [Number\[\]](#) - Pixel shift along the axes at the given step.
- wayPoint: [multiRouter.WayPoint](#) - A reference to the waypoint object being dragged.

### waypointdragend

End of waypoint dragging. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- wayPoint: [multiRouter.WayPoint](#) - A reference to the waypoint object being dragged.

### waypointdragstart

Start of waypoint dragging. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- domEvent: [DomEvent](#) - The source DOM event, if there is one.
- wayPoint: [multiRouter.WayPoint](#) - A reference to the waypoint object being dragged.

### waypointremove

Deleting a waypoint. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- wayPoint: [multiRouter.WayPoint](#) - A reference to the deleted waypoint object.

## Methods details

### isActive

```
{Boolean} isActive()
```

**Returns** a flag for whether the editor is currently enabled.

### start

```
{ } start(state)
```

Enables the editor.

#### Parameters:

Parameter	Default value	Description
<a href="#">state</a> *	—	Type: Object The initial state of the editor.

\* Mandatory parameter/option.

### stop

```
{ } stop()
```

Disables the editor.

## multiRouter.masstransit

### multiRouter.masstransit.Path

**Note:** The constructor of the `multiRouter.masstransit.Path` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeoObject](#).

Representation of the route path for public transport. A single route can contain several paths, and each path connects two waypoints.

[Fields](#) | [Events](#) | [Methods](#)

Creates a representation of the public transport route path.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">model</a>	<a href="#">multiRouter.masstransit.PathModel</a>	Data model for the multi-stop route's path.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">data.Manager</a>	Multi-stop route's path data. The following fields are available: <ul style="list-style-type: none"> <li>• <code>index</code>: Integer - The sequential number of the path in the multi-stop route's corresponding route.</li> <li>• <code>type</code>: String - Route type identifier, which takes the value "masstransit" for public transport routes.</li> <li>• <code>distance</code>: Object - An object with the "text" and "value" fields that describes the length of the path in meters.</li> <li>• <code>duration</code>: Object - An object with the "text" and "value" fields that describes the travel time of the path in seconds.</li> <li>• <code>coordinates</code>: Number[][] - Coordinates of all points on the path.</li> <li>• <code>encodedCoordinates</code>: String - A string of base64-encoded coordinates for all points on the path.</li> </ul>
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

### Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometrychange</a>	Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the geometry.</li> </ul> Inherited from <a href="#">IGeoObject</a> .

Name	Description
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldMap</code> - Old map.</li><li>• <code>newMap</code> - New map.</li></ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li><code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li><code>oldParent</code> - Old parent.</li> <li><code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">update</a>	<p>Updating the path rendering. Instance of the <a href="#">Event</a> class.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	<p>Returns reference to the map.</p> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	<p>Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>

Name	Returns	Description
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay.  Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IChildOnMap</a> .
<a href="#">getSegmentMarkers()</a>	<a href="#">GeoObjectCollection</a>	Returns a child collection of segment markers.
<a href="#">getSegments()</a>	<a href="#">GeoObjectCollection</a>	Returns the child collection of segments that the path consists of.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .

## Fields details

### model

```
{multiRouter.masstransit.PathModel} model
```

Data model for the multi-stop route's path.

### properties

```
{data.Manager} properties
```

Multi-stop route's path data. The following fields are available:

- **index**: Integer - The sequential number of the path in the multi-stop route's corresponding route.
- **type**: String - Route type identifier, which takes the value "masstransit" for public transport routes.
- **distance**: Object - An object with the "text" and "value" fields that describes the length of the path in meters.
- **duration**: Object - An object with the "text" and "value" fields that describes the travel time of the path in seconds.
- **coordinates**: Number[][] - Coordinates of all points on the path.
- **encodedCoordinates**: String - A string of base64-encoded coordinates for all points on the path.

## Events details

### update

Updating the path rendering. Instance of the [Event](#) class.

## Methods details

### getSegmentMarkers

```
{GeoObjectCollection} getSegmentMarkers()
```

**Returns** a child collection of segment markers.

## getSegments

```
{GeoObjectCollection} getSegments()
```

**Returns** the child collection of segments that the path consists of.

### multiRouter.masstransit.PathModel

**Note:** The constructor of the `multiRouter.masstransit.PathModel` class is hidden, as this class is not intended for autonomous initialization.

Extends [IEventEmitter](#).

Data model of a path on a public transport route. A single route can contain several paths, and each path connects two waypoints.

[Fields](#) | [Events](#) | [Methods](#)

Creates a data model for a path on a public transport route.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">properties</a>	<a href="#">data.Manager</a>	Multi-stop route's path data. The following fields are available: <ul style="list-style-type: none"> <li><code>index</code>: Integer - The sequential number of the path in the multi-stop route's corresponding route.</li> <li><code>type</code>: String - Route type identifier, which takes the value "masstransit" for public transport routes.</li> <li><code>distance</code>: Object - An object with the "text" and "value" fields that describes the length of the path in meters.</li> <li><code>duration</code>: Object - An object with the "text" and "value" fields that describes the travel time of the path in seconds.</li> <li><code>coordinates</code>: Number[][] - Coordinates of all points on the path.</li> <li><code>encodedCoordinates</code>: String - A string of base64-encoded coordinates for all points on the path.</li> </ul>
<a href="#">route</a>	<a href="#">multiRouter.masstransit.RouteModel</a>	Reference to the parent model of the route.

### Events

Name	Description
<a href="#">update</a>	Updating the model with new data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>segmentsChange</code>: Boolean - Flag for whether the set of segments is changed</li> </ul>

### Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destroys a model.
<a href="#">getSegments()</a>	( <a href="#">multiRouter.masstransit.TransportModel</a>   <a href="#">multiRouter.masstransit.TransportModel</a>   <a href="#">multiRouter.masstransit.WalkSegmentModel</a> ) []	Returns an array of path segments

Name	Returns	Description
<code>getType()</code>	String	Returns ID of the route type. For public transport routes, the string "masstransit" is returned.
<code>update(pathJson)</code>		Updates the state of the model.

## Fields details

### properties

```
{data.Manager} properties
```

Multi-stop route's path data. The following fields are available:

- `index`: Integer - The sequential number of the path in the multi-stop route's corresponding route.
- `type`: String - Route type identifier, which takes the value "masstransit" for public transport routes.
- `distance`: Object - An object with the "text" and "value" fields that describes the length of the path in meters.
- `duration`: Object - An object with the "text" and "value" fields that describes the travel time of the path in seconds.
- `coordinates`: Number[][] - Coordinates of all points on the path.
- `encodedCoordinates`: String - A string of base64-encoded coordinates for all points on the path.

### route

```
{multiRouter.masstransit.RouteModel} route
```

Reference to the parent model of the route.

## Events details

### update

Updating the model with new data. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `segmentsChange`: Boolean - Flag for whether the set of segments is changed

## Methods details

### destroy

```
{ } destroy()
```

Destroys a model.

### getSegments

```
{ (multiRouter.masstransit.TransferSegmentModel |  
multiRouter.masstransit.TransportSegmentModel |  
multiRouter.masstransit.WalkSegmentModel) [] } getSegments()
```

Returns array of path segments.

### getType

```
{String} getType()
```

Returns ID of the route type. For public transport routes, the string "masstransit" is returned.

**update**

```
{ } update (pathJson)
```

Updates the state of the model.

**Parameters:**

Parameter	Default value	Description
<a href="#">pathJson</a> *	—	Type: Object JSON data.

\* Mandatory parameter/option.

**multiRouter.masstransit.Route**

**Note:** The constructor of the `multiRouter.masstransit.Route` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeoObject](#).

Displaying an individual route of public transport. A multi-stop route can consist of several individual routes.

[Fields](#) | [Events](#) | [Methods](#)

Creates the representation to display an individual route of public transport.

**Fields**

Name	Type	Description
<a href="#">balloon</a>	<a href="#">IMultiRouterRouteBalloon</a>	Balloon of a route.
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">model</a>	<a href="#">multiRouter.masstransit.RouteModel</a>	The data model of an individual route.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">data.Manager</a>	The route data. The following fields are available: <ul style="list-style-type: none"> <li><code>index</code>: Integer - The ordinal number of the route in a multi-stop route.</li> <li><code>type</code>: String - Route type identifier, which takes the value "masstransit" for public transport routes.</li> <li><code>distance</code>: Object - An object with the "text" and "value" fields that specifies the length of the route in meters.</li> <li><code>duration</code>: Object - An object with the "text" and "value" fields that specifies the travel time of the route in seconds.</li> <li><code>boundedBy</code>: Number[][] - Coordinates of the upper and lower corners of the rectangle that bounds the route.</li> </ul>
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

## Events

Name	Description
<a href="#">balloonclose</a>	Closing the balloon.
<a href="#">balloonopen</a>	Opening the balloon.
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">update</a>	<p>Updating the route rendering. Instance of the <a href="#">Event</a> class.</p>

Name	Description
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	<p>Returns reference to the map.</p> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	<p>Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	<p>The method provides synchronous access to the overlay.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	<p>Returns link to the parent object, or null if the parent element was not set.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>
<a href="#">getPaths()</a>	<a href="#">GeoObjectCollection</a>	<p>Returns a child collection of paths that make up the route.</p>
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	<p>Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>

## Fields details

### balloon

```
{IMultiRouterRouteBalloon} balloon
```

Balloon of a route.

### model

```
{multiRouter.masstransit.RouteModel} model
```

The data model of an individual route.

### properties

```
{data.Manager} properties
```

The route data. The following fields are available:

- **index:** Integer - The ordinal number of the route in a multi-stop route.
- **type:** String - Route type identifier, which takes the value "masstransit" for public transport routes.
- **distance:** Object - An object with the "text" and "value" fields that specifies the length of the route in meters.
- **duration:** Object - An object with the "text" and "value" fields that specifies the travel time of the route in seconds.
- **boundedBy:** Number[][] - Coordinates of the upper and lower corners of the rectangle that bounds the route.

### Events details

#### balloonclose

Closing the balloon.

#### balloonopen

Opening the balloon.

#### update

Updating the route rendering. Instance of the [Event](#) class.

### Methods details

#### getPaths

```
{GeoObjectCollection} getPaths()
```

**Returns** a child collection of paths that make up the route.

#### multiRouter.masstransit.RouteModel

**Note:** The constructor of the multiRouter.masstransit.RouteModel class is hidden, as this class is not intended for autonomous initialization.

Extends [IEventEmitter](#).

Data model of an individual route of public transport. A multi-stop route can consist of several individual routes.

[Fields](#) | [Events](#) | [Methods](#)

Creates the data model for an individual route of public transport.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">multiRoute</a>	<a href="#">multiRouter.MultiRouteModel</a>	Reference to the parent model of a multi-stop route.
<a href="#">properties</a>	<a href="#">data.Manager</a>	The route data. The following fields are available: <ul style="list-style-type: none"> <li>• <b>index:</b> Integer - The ordinal number of the route in a multi-stop route.</li> <li>• <b>type:</b> String - Route type identifier, which takes the value "masstransit" for public transport routes.</li> <li>• <b>distance:</b> Object - An object with the "text" and "value" fields that specifies the length of the route in meters.</li> <li>• <b>duration:</b> Object - An object with the "text" and "value" fields that specifies the travel time of the route in seconds.</li> <li>• <b>boundedBy:</b> Number[][] - Coordinates of the upper and lower corners of the rectangle that bounds the route.</li> </ul>

## Events

Name	Description
<a href="#">update</a>	<p>Updating the model with new data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>pathsChange</code>: Boolean - Flag for whether the set of paths is changed.</li> </ul>

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destroys a model.
<a href="#">getPaths()</a>	<a href="#">multiRouter.masstransit.PathModel</a>	Returns array of route paths.
<a href="#">getType()</a>	String	Returns ID of the route type. For public transport routes, the string "masstransit" is returned.
<a href="#">update(routeJson)</a>		Updates the state of the model.

## Fields details

### multiRoute

```
{multiRouter.MultiRouteModel} multiRoute
```

Reference to the parent model of a multi-stop route.

### properties

```
{data.Manager} properties
```

The route data. The following fields are available:

- `index`: Integer - The ordinal number of the route in a multi-stop route.
- `type`: String - Route type identifier, which takes the value "masstransit" for public transport routes.
- `distance`: Object - An object with the "text" and "value" fields that specifies the length of the route in meters.
- `duration`: Object - An object with the "text" and "value" fields that specifies the travel time of the route in seconds.
- `boundedBy`: Number[][] - Coordinates of the upper and lower corners of the rectangle that bounds the route.

## Events details

### update

Updating the model with new data. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `pathsChange`: Boolean - Flag for whether the set of paths is changed.

## Methods details

### destroy

```
{ } destroy()
```

Destroys a model.

**getPaths**

```
{multiRouter.masstransit.PathModel[]} getPaths()
```

**Returns** array of route paths.

**getType**

```
{String} getType()
```

**Returns** ID of the route type. For public transport routes, the string "masstransit" is returned.

**update**

```
{ } update(routeJson)
```

Updates the state of the model.

**Parameters:**

Parameter	Default value	Description
<a href="#">routeJson</a> *	—	Type: Object JSON data.

\* Mandatory parameter/option.

**multiRouter.masstransit.StopModel**

**Note:** The constructor of the multiRouter.masstransit.StopModel class is hidden, as this class is not intended for autonomous initialization.

Extends [IEventEmitter](#).

Data model for a stop on a transportation segment of a public transport route.

[Fields](#) | [Events](#) | [Methods](#)

Creates the data model for a stop of a transport segment.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">geometry.base.Point</a>	Geometry of the stop.
<a href="#">properties</a>	<a href="#">data.Manager</a>	Stop data. The following fields are available: <ul style="list-style-type: none"> <li>Integer - The ordinal number of the stop in the set of stops corresponding to the transit segment.</li> <li>id: String - The stop ID.</li> <li>name: String - The name of the stop.</li> <li>lodIndex: Integer - The ordinal number of the stop point in the full set of coordinates of the corresponding route path.</li> </ul>
<a href="#">segment</a>	<a href="#">multiRouter.masstransit.TransportSegmentModel</a>	Reference to the parent model of the transit segment.

**Events**

Name	Description
<a href="#">update</a>	Updating the model with new data. Instance of the <a href="#">Event</a> class.

**Methods**

Name	Description
<a href="#">update(stopJson)</a>	Updates the state of the model.

**Fields details****geometry**

```
{geometry.base.Point} geometry
```

Geometry of the stop.

**properties**

```
{data.Manager} properties
```

Stop data. The following fields are available:

- Integer - The ordinal number of the stop in the set of stops corresponding to the transit segment.
- id: String - The stop ID.
- name: String - The name of the stop.
- lodIndex: Integer - The ordinal number of the stop point in the full set of coordinates of the corresponding route path.

**segment**

```
{multiRouter.masstransit.TransportSegmentModel} segment
```

Reference to the parent model of the transit segment.

**Events details****update**

Updating the model with new data. Instance of the [Event](#) class.

**Methods details****update**

```
{ } update (stopJson)
```

Updates the state of the model.

**Parameters:**

Parameter	Default value	Description
<a href="#">stopJson</a> *	—	Type: Object JSON data.

\* Mandatory parameter/option.

**multiRouter.masstransit.TransferSegment**

**Note:** The constructor of the multiRouter.masstransit.TransferSegment class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeoObject](#).

Representation of a transfer segment on a public transport route. A segment of a path on a public transport route is part of a path from one stop to another.

[Fields](#) | [Events](#) | [Methods](#)

Creates a representation to display a transfer segment on a public transport route.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">model</a>	<a href="#">multiRouter.masstransit.TransferSegmentModel</a>	Data model for a segment.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">data.Manager</a>	Segment data. The following fields are available: <ul style="list-style-type: none"> <li>• index: Integer - The ordinal number of the segment in the array of segments of the corresponding route path.</li> <li>• type: String - Segment type identifier, which takes the value "transfer" for transfer segments.</li> <li>• text: String - Text description of the segment.</li> <li>• distance: Object - An object with the "text" and "value" fields that specifies the length of the segment in meters.</li> <li>• duration: Object - An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.</li> <li>• lodIndex: Integer - The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.</li> </ul>
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

**Events**

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li></ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>oldMap - Old map.</li><li>newMap - New map.</li></ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">update</a>	<p>Updating the segment rendering. Instance of the <code>Event</code> class.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IChildOnMap</a> .

**Fields details****model**

```
{multiRouter.masstransit.TransferSegmentModel} model
```

Data model for a segment.

**properties**

```
{data.Manager} properties
```

Segment data. The following fields are available:

- **index:** Integer - The ordinal number of the segment in the array of segments of the corresponding route path.
- **type:** String - Segment type identifier, which takes the value "transfer" for transfer segments.
- **text:** String - Text description of the segment.
- **distance:** Object - An object with the "text" and "value" fields that specifies the length of the segment in meters.
- **duration:** Object - An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.
- **lodIndex:** Integer - The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

## Events details

### update

Updating the segment rendering. Instance of the [Event](#) class.

### multiRouter.masstransit.TransferSegmentModel

**Note:** The constructor of the `multiRouter.masstransit.TransferSegmentModel` class is hidden, as this class is not intended for autonomous initialization.

Extends [IEventEmitter](#).

Data model for a transfer segment of a path on a public transport route. A segment of a path on a public transport route is part of a path from one stop to another.

[Fields](#) | [Events](#) | [Methods](#)

Creates the data model for a transfer segment of a path on a public transport route.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">geometry.base.LineString</a>	Geometry of a segment.
<a href="#">path</a>	<a href="#">multiRouter.masstransit.PathModel</a>	Reference to the parent model of the path.
<a href="#">properties</a>	<a href="#">data.Manager</a>	Segment data. The following fields are available: <ul style="list-style-type: none"> <li>index: Integer - The ordinal number of the segment in the array of segments of the corresponding route path.</li> <li>type: String - Segment type identifier, which takes the value "transfer" for transfer segments.</li> <li>text: String - Text description of the segment.</li> <li>distance: Object - An object with the "text" and "value" fields that specifies the length of the segment in meters.</li> <li>duration: Object - An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.</li> <li>lodIndex: Integer - The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.</li> </ul>

## Events

Name	Description
<a href="#">update</a>	Updating the model with new data. Instance of the <a href="#">Event</a> class.

## Methods

Name	Returns	Description
<a href="#">destroy(segmentJson)</a>		Updates the state of the model.
<a href="#">getType()</a>	String	Returns ID of the segment type. For transfer segments on public transport routes, it returns the string "transfer".

## Fields details

### geometry

```
{geometry.base.LineString} geometry
```

Geometry of a segment.

### path

```
{multiRouter.masstransit.PathModel} path
```

Reference to the parent model of the path.

## properties

```
{data.Manager} properties
```

Segment data. The following fields are available:

- **index**: Integer - The ordinal number of the segment in the array of segments of the corresponding route path.
- **type**: String - Segment type identifier, which takes the value "transfer" for transfer segments.
- **text**: String - Text description of the segment.
- **distance**: Object - An object with the "text" and "value" fields that specifies the length of the segment in meters.
- **duration**: Object - An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.
- **lodIndex**: Integer - The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

## Events details

### update

Updating the model with new data. Instance of the [Event](#) class.

## Methods details

### destroy

```
{ } destroy(segmentJson)
```

Updates the state of the model.

### Parameters:

Parameter	Default value	Description
<a href="#">segmentJson</a> *	—	Type: Object JSON data.

\* Mandatory parameter/option.

### getType

```
{String} getType()
```

**Returns** ID of the segment type. For transfer segments on public transport routes, it returns the string "transfer".

**multiRouter.masstransit.TransportSegment**

**Note:** The constructor of the `multiRouter.masstransit.TransportSegment` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeoObject](#).

Representation of a transport segment on a public transport route. A segment of a path on a public transport route is part of a path from one stop to another.

[Fields](#) | [Events](#) | [Methods](#)

Creates the representation to display a transport segment on a public transport route.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">model</a>	<a href="#">multiRouter.masstransit.TransportSegmentModel</a>	Data model for a segment.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">data.Manager</a>	Segment data. The following fields are available: <ul style="list-style-type: none"> <li>index: Integer - The ordinal number of the segment in the array of segments of the corresponding route path.</li> <li>String - Segment type identifier, which takes the value "transport" for transport segments.</li> <li>text: String - Text description of the segment.</li> <li>transports: <a href="#">ITransportProperties[]</a> - An array describing the specific forms of transport available for traveling on this segment.</li> <li>stops: Object - Description of stops in the format <a href="#">GeoJson.FeatureCollection</a>.</li> <li>distance: Object - An object with the "text" and "value" fields that specifies the length of the segment in meters.</li> <li>duration: Object - An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.</li> <li>lodIndex: Integer - The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.</li> </ul>
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

**Events**

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li></ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>oldMap - Old map.</li><li>newMap - New map.</li></ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">update</a>	<p>Updating the segment rendering. Instance of the <code>Event</code> class.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IChildOnMap</a> .

**Fields details****model**

```
{multiRouter.masstransit.TransportSegmentModel} model
```

Data model for a segment.

**properties**

```
{data.Manager} properties
```

Segment data. The following fields are available:

- **index**: Integer - The ordinal number of the segment in the array of segments of the corresponding route path.
- **String** - Segment type identifier, which takes the value "transport" for transport segments.
- **text**: String - Text description of the segment.
- **transports**: [ITransportProperties\[\]](#) - An array describing the specific forms of transport available for traveling on this segment.
- **stops**: Object - Description of stops in the format [GeoJson:FeatureCollection](#).
- **distance**: Object - An object with the "text" and "value" fields that specifies the length of the segment in meters.
- **duration**: Object - An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.
- **lodIndex**: Integer - The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

## Events details

### update

Updating the segment rendering. Instance of the [Event](#) class.

### multiRouter.masstransit.TransportSegmentModel

**Note:** The constructor of the `multiRouter.masstransit.TransportSegmentModel` class is hidden, as this class is not intended for autonomous initialization.

Extends [IEventEmitter](#).

Data model for a transportation segment of a path on a public transport route. A segment of a path on a public transport route is part of a path from one stop to another.

[Fields](#) | [Events](#) | [Methods](#)

Creates a data model for a transportation segment of a path on a public transport route.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">geometry.base.LineString</a>	Geometry of a segment.
<a href="#">path</a>	<a href="#">multiRouter.masstransit.PathModel</a>	Reference to the parent model of the path.
<a href="#">properties</a>	<a href="#">data.Manager</a>	Segment data. The following fields are available: <ul style="list-style-type: none"> <li>index: Integer - The ordinal number of the segment in the array of segments of the corresponding route path.</li> <li>String - Segment type identifier, which takes the value "transport" for transport segments.</li> <li>text: String - Text description of the segment.</li> <li>transports: <a href="#">ITransportProperties[]</a> - An array describing the specific forms of transport available for traveling on this segment.</li> <li>stops: Object - Description of stops in the format <a href="#">GeoJson:FeatureCollection</a>.</li> <li>distance: Object - An object with the "text" and "value" fields that specifies the length of the segment in meters.</li> <li>duration: Object - An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.</li> <li>lodIndex: Integer - The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.</li> </ul>

## Events

Name	Description
<a href="#">update</a>	Updating the model with new data. Instance of the <a href="#">Event</a> class.

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destroys a model.
<a href="#">getStops()</a>	<a href="#">multiRouter.masstransit.StopModel[]</a>	Returns array of stops on a transport route.

Name	Returns	Description
<a href="#">getType()</a>	String	Returns ID of the segment type. For transport segments on public transport routes, it returns the string "transport".
<a href="#">update(segmentJson)</a>		Updates the state of the model.

## Fields details

### geometry

```
{geometry.base.LineString} geometry
```

Geometry of a segment.

### path

```
{multiRouter.masstransit.PathModel} path
```

Reference to the parent model of the path.

### properties

```
{data.Manager} properties
```

Segment data. The following fields are available:

- index: Integer - The ordinal number of the segment in the array of segments of the corresponding route path.
- String - Segment type identifier, which takes the value "transport" for transport segments.
- text: String - Text description of the segment.
- transports: [ITransportProperties\[\]](#) - An array describing the specific forms of transport available for traveling on this segment.
- stops: Object - Description of stops in the format [GeoJson.FeatureCollection](#).
- distance: Object - An object with the "text" and "value" fields that specifies the length of the segment in meters.
- duration: Object - An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.
- lodIndex: Integer - The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

## Events details

### update

Updating the model with new data. Instance of the [Event](#) class.

## Methods details

### destroy

```
{ } destroy()
```

Destroys a model.

### getStops

```
{multiRouter.masstransit.StopModel[]} getStops()
```

**Returns** array of stops on a transport route.

### getType

```
{String} getType()
```

**Returns** ID of the segment type. For transport segments on public transport routes, it returns the string "transport".

### update

```
{ } update(segmentJson)
```

Updates the state of the model.

#### Parameters:

Parameter	Default value	Description
<a href="#">segmentJson</a> *	—	Type: Object JSON data.

\* Mandatory parameter/option.

### multiRouter.masstransit.WalkSegment

**Note:** The constructor of the `multiRouter.masstransit.WalkSegment` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeoObject](#).

Representation of a walking segment on a public transport route. A segment of a path on a public transport route is part of a path from one stop to another.

[Fields](#) | [Events](#) | [Methods](#)

Creates the representation to display a walking segment on a public transport route.

#### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">model</a>	<a href="#">multiRouter.masstransit.WalkSegmentModel</a>	Data model for a segment.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

Name	Type	Description
<a href="#">properties</a>	<a href="#">data.Manager</a>	<p>Segment data. The following fields are available:</p> <ul style="list-style-type: none"> <li>index: Integer - The ordinal number of the segment in the array of segments of the corresponding route path.</li> <li>type: String - Segment type identifier, which takes the value "walk" for walking segments.</li> <li>text: String - Text description of the segment.</li> <li>distance: Object - An object with the "text" and "value" fields that specifies the length of the segment in meters.</li> <li>duration: Object - An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.</li> <li>lodIndex: Integer - The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.</li> </ul>
<a href="#">state</a>	<a href="#">IDataManager</a>	<p>State of the geo object.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>

## Events

Name	Description
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>

Name	Description
<a href="#">parentchange</a>	The parent object reference changed. Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .
<a href="#">propertieschange</a>	Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> Inherited from <a href="#">IGeoObject</a> .
<a href="#">update</a>	Updating the segment rendering. Instance of the <a href="#">Event</a> class.
<a href="#">wheel</a>	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IChildOnMap</a> .

## Fields details

### model

```
{multiRouter.masstransit.WalkSegmentModel} model
```

Data model for a segment.

### properties

```
{data.Manager} properties
```

Segment data. The following fields are available:

- **index:** Integer - The ordinal number of the segment in the array of segments of the corresponding route path.
- **type:** String - Segment type identifier, which takes the value "walk" for walking segments.
- **text:** String - Text description of the segment.
- **distance:** Object - An object with the "text" and "value" fields that specifies the length of the segment in meters.
- **duration:** Object - An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.
- **lodIndex:** Integer - The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

### Events details

#### update

Updating the segment rendering. Instance of the [Event](#) class.

#### multiRouter.masstransit.WalkSegmentModel

**Note:** The constructor of the multiRouter.masstransit.WalkSegmentModel class is hidden, as this class is not intended for autonomous initialization.

Extends [IEventEmitter](#).

Data model for a walking segment of a path on a public transport route. A segment of a path on a public transport route is part of a path from one stop to another.

[Fields](#) | [Events](#) | [Methods](#)

Creates the data model for a walking segment of a path on a public transport route.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">geometry.base.LineString</a>	Geometry of a segment.
<a href="#">path</a>	<a href="#">multiRouter.masstransit.PathModel</a>	Reference to the parent model of the path.
<a href="#">properties</a>	<a href="#">data.Manager</a>	Segment data. The following fields are available: <ul style="list-style-type: none"> <li>• <b>index:</b> Integer - The ordinal number of the segment in the array of segments of the corresponding route path.</li> <li>• <b>type:</b> String - Segment type identifier, which takes the value "walk" for walking segments.</li> <li>• <b>text:</b> String - Text description of the segment.</li> <li>• <b>distance:</b> Object - An object with the "text" and "value" fields that specifies the length of the segment in meters.</li> <li>• <b>duration:</b> Object - An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.</li> <li>• <b>lodIndex:</b> Integer - The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.</li> </ul>

## Events

Name	Description
<a href="#">update</a>	Updating the model with new data. Instance of the <a href="#">Event</a> class.

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destroys a model.
<a href="#">getType()</a>	String	Returns ID of the segment type. For walking segments on public transport routes, it returns the string "walk".

## Fields details

### geometry

```
{geometry.base.LineString} geometry
```

Geometry of a segment.

### path

```
{multiRouter.masstransit.PathModel} path
```

Reference to the parent model of the path.

### properties

```
{data.Manager} properties
```

Segment data. The following fields are available:

- **index:** Integer - The ordinal number of the segment in the array of segments of the corresponding route path.
- **type:** String - Segment type identifier, which takes the value "walk" for walking segments.
- **text:** String - Text description of the segment.
- **distance:** Object - An object with the "text" and "value" fields that specifies the length of the segment in meters.
- **duration:** Object - An object with the "text" and "value" fields that specifies the travel time of the segment in seconds.
- **lodIndex:** Integer - The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

## Events details

### update

Updating the model with new data. Instance of the [Event](#) class.

## Methods details

### destroy

```
{ } destroy()
```

Destroys a model.

## getType

```
{String} getType()
```

**Returns** ID of the segment type. For walking segments on public transport routes, it returns the string "walk".

## multiRouter.MultiRoute

Extends [IGeoObject](#).

Multi-route on the map. Displays the route on the map along with one or more alternatives. Provides the interface for selecting the active route.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
multiRouter.MultiRoute(model[, options])
```

Creates a multi-route on the map.

#### Parameters:

Parameter	Default value	Description
<a href="#">model</a> *	—	Type: <a href="#">multiRouter.MultiRouteModel</a>    <a href="#">IMultiRouteModelJson</a>  The data model of a multi-route, or the model description object. If you pass a description object, the model (which is based on it) is created automatically.

Parameter	Default value	Description
<a href="#">options</a>	—	<p>Type: Object</p> <p>Multi-route options. To define options for the parts of a multi-route, use the following prefixes:</p> <ul style="list-style-type: none"> <li><code>wayPoint</code> - Waypoint options.</li> <li><code>wayPointStart</code> - Display options for the starting waypoint.</li> <li><code>wayPointFinish</code> - Display options for the end waypoint.</li> <li><code>viaPoint</code> - Throughpoint options.</li> <li><code>pin</code> - Options for point markers on the route.</li> <li><code>editor</code> - Options for the multi-route editor (see <a href="#">multiRouter.Editor</a>).</li> </ul> <p>To define options for individual routes within a multi-route, use the following prefixes:</p> <ul style="list-style-type: none"> <li><code>route</code> - Options for routes, including inactive ones.</li> <li><code>routeActive</code> - Options for the active route.</li> </ul> <p>Note that options specified with the "routeActive" prefix have a higher priority than the options with the "route" prefix. To define options for segments of public transport routes, use the following prefixes:</p> <ul style="list-style-type: none"> <li><code>routeMarker</code> - Options for route segment markers.</li> <li><code>routeWalkMarker</code> - Options for markers of walking segments on the route.</li> <li><code>routeTransferMarker</code> - Options for transfer segment markers.</li> <li><code>routeTransportMarker</code> - Options for transit segment markers on the route.</li> <li><code>routeWalkSegment</code> - Options for walking segment lines on the route.</li> <li><code>routeTransferSegment</code> - Options for transfer segment lines.</li> <li><code>routeTransportSegment</code> - Options for transit segment lines on the route.</li> <li><code>routePedestrianSegment</code> - Options for pedestrian route segment lines.</li> </ul> <p>These prefixes are also available in the "routeActive*" version.</p>
<a href="#">options.activeRouteAutoSelection</a>	true	<p>Type: Boolean</p> <p>After the data is refreshed, automatically use the route with the shortest travel time as the active route.</p>

Parameter	Default value	Description
<a href="#">options.boundsAutoApply</a>	false	Type: Boolean  When adding a multi-route to the map, you can automatically set the center and the zoom level so that the multi-route is entirely visible.
<a href="#">options.dragUpdateInterval</a>	—	Type: String Number  Time interval of rebuilding the route while reference points are being dragged. Can be set in milliseconds, or the optimal value can be calculated automatically. The value of this option is translated to the <a href="#">IMultiRouteParams.requestSendInterval</a> parameter of the multi-route.
<a href="#">options.preventDragUpdate</a>	false	Type: Boolean  Allows to disable the route update while reference points are being dragged.
<a href="#">options.useMapMargin</a>	true	Type: Boolean  Whether to account for map margins <a href="#">map.margin.Manager</a> .
<a href="#">options.zoomMargin</a>	0	Type: Number Number[]  Offset from the map viewport borders when changing the zoom level. If a single number is set, it is applied to each side. If two numbers are set, they are the horizontal and vertical margins, respectively. If an array of four numbers is set, they are the top, right, bottom, and left margins. When the "useMapMargin" option is enabled, the "zoomMargin" value is combined with the values that were calculated in the margins manager <a href="#">map.margin.Manager</a> .

\* Mandatory parameter/option.

### Examples:

#### 1.

```
// Creating a multi-route and adding it to the map.
var multiRoute = new ymaps.multiRouter.MultiRoute({
  referencePoints: ['Moscow, Leninsky Prospekt', 'Moscow, Kulakov pereulok'],
}, {
  editorDrawOver: false,
  wayPointDraggable: true,
  viaPointDraggable: true,
  // Setting a custom design for multi-route lines.
  routeStrokeColor: "000088",
  routeActiveStrokeColor: "ff0000",
  pinIconFillColor: "ff0000",
  boundsAutoApply: true,
  zoomMargin: 30
});
myMap.geoObjects.add(multiRoute);
```

## 2.

```

// Creating a multi-route and adding it to the map.
var multiRoute = new ymaps.multiRouter.MultiRoute({
  referencePoints: ['Moscow, Leninsky Prospekt', 'Moscow, Kulakov pereulok', 'Zelenograd'],
});
myMap.geoObjects.add(multiRoute);

// Once multi-route is loaded.
multiRoute.events.once('update', function () {
  // Set first non-blocked route as active and open it's balloon.
  var routes = multiRoute.getRoutes();
  for (var i = 0, l = routes.getLength(); i < l; i++) {
    var route = routes.get(i);
    if (!route.properties.get('blocked')) {
      multiRoute.setActiveRoute(route);
      route.balloon.open();
      break;
    }
  }
});

```

## Fields

Name	Type	Description
<a href="#">editor</a>	<a href="#">multiRouter.EditorAdapter</a>	Multi-stop route editor.
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">model</a>	<a href="#">multiRouter.MultiRouteModel</a>	The data model of a multi-route.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">IDataManager</a>	Geo object data. Inherited from <a href="#">IGeoObject</a> .
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

## Events

Name	Description
<a href="#">activeroutechange</a>	Change to the active route. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>oldActiveRoute</code>: <a href="#">multiRouter.driving.Route</a> <a href="#">multiRouter.masstransit.Route</a> null - previous active route.</li> </ul>
<a href="#">balloonclose</a>	Closing the balloon. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>originalTarget</code>: <a href="#">multiRouter.pedestrian.Route</a> <a href="#">multiRouter.driving.Route</a> <a href="#">multiRouter.masstransit.Route</a> null - route on which balloon was closed.</li> </ul>
<a href="#">balloonopen</a>	Opening the balloon. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>originalTarget</code>: <a href="#">multiRouter.pedestrian.Route</a> <a href="#">multiRouter.driving.Route</a> <a href="#">multiRouter.masstransit.Route</a> null - route on which balloon was opened.</li> </ul>

Name	Description
<a href="#">boundsautoapply</a>	The event occurs at the time of setting the map center and its zoom level for optimal display of the multi-route. Also see the description of the <code>boundsAutoApply</code> option. Instance of the <a href="#">Event</a> class.
<a href="#">boundschange</a>	Changing coordinates of the geographical area covering the multi-route. Instance of the <a href="#">Event</a> class.
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometrychange</a>	Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"><li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the geometry.</li></ul> Inherited from <a href="#">IGeoObject</a> .
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"><li>• <code>oldMap</code> - Old map.</li><li>• <code>newMap</code> - New map.</li></ul> Inherited from <a href="#">IParentOnMap</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseleave</a>	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mousemove</a>	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseup</a>	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">pixelboundschange</a>	<p>Changing pixel coordinates of the area covering the multi-route. Instance of the <a href="#">Event</a> class.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">update</a>	<p>Updating the multi-route. Instance of the <a href="#">Event</a> class.</p>

Name	Description
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getActiveRoute()</a>	<a href="#">multiRouter.driving.Route</a>   <a href="#">multiRouter.ActiveRoute</a>  null	Returns active route.
<a href="#">getBounds()</a>	Number[][] null	Returns the geographical coordinates of the area covering the multi-route.
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">getPixelBounds()</a>	Number[][] null	Returns the global pixel coordinates of the area covering the multi-route.
<a href="#">getRoutes()</a>	<a href="#">GeoObjectCollection</a>	Returns child collection of individual routes on the multi-route.
<a href="#">getViaPoints()</a>	<a href="#">GeoObjectCollection</a>	Returns child collection of throughpoints for the multi-route.
<a href="#">getWayPoints()</a>	<a href="#">GeoObjectCollection</a>	Returns child collection of waypoints for the multi-route.
<a href="#">setActiveRoute(route)</a>		Sets the active route. The previous active route is deactivated and the <a href="#">multiRouter.MultiRoute.event:activeroutechange</a> event is generated.

Name	Returns	Description
<code>setParent(parent)</code>	<code>IChildOnMap</code>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <code>IChildOnMap</code> .

## Fields details

### editor

```
{multiRouter.EditorAddon} editor
```

Multi-stop route editor.

### Example:

```
// Start of route editing.
ymaps.route(['Moscow', 'Petersburg'], { multiRoute: true })
  .done(function (multiRoute) {
    myMap.geoObjects.add(multiRoute);
    multiRoute.editor.start({
      addWayPoints: true,
      removeWayPoints: true
    });
    // ...
    // End of route editing.
    multiRoute.editor.stop();
  });
```

### model

```
{multiRouter.MultiRouteModel} model
```

The data model of a multi-route.

## Events details

### activeroutechange

Change to the active route. Instance of the `Event` class. Names of fields that are available via the `Event.get` method:

- `oldActiveRoute`: `multiRouter.driving.Route|multiRouter.masstransit.Route|null` - previous active route.

### balloonclose

Closing the balloon. Names of fields that are available via the `Event.get` method:

- `originalTarget`: `multiRouter.pedestrian.Route|multiRouter.driving.Route|multiRouter.masstransit.Route|null` - route on which balloon was closed.

### balloonopen

Opening the balloon. Names of fields that are available via the `Event.get` method:

- `originalTarget`: `multiRouter.pedestrian.Route|multiRouter.driving.Route|multiRouter.masstransit.Route|null` - route on which balloon was opened.

### boundsautoapply

The event occurs at the time of setting the map center and its zoom level for optimal display of the multi-route. Also see the description of the `boundsAutoApply` option. Instance of the `Event` class.

### boundschange

Changing coordinates of the geographical area covering the multi-route. Instance of the [Event](#) class.

### pixelboundschange

Changing pixel coordinates of the area covering the multi-route. Instance of the [Event](#) class.

### update

Updating the multi-route. Instance of the [Event](#) class.

## Methods details

### getActiveRoute

```
{multiRouter.driving.Route|multiRouter.masstransit.Route|null} getActiveRoute()
```

**Returns** active route.

### getBounds

```
{Number[][]|null} getBounds()
```

**Returns** the geographical coordinates of the area covering the multi-route.

### getPixelBounds

```
{Number[][]|null} getPixelBounds()
```

**Returns** the global pixel coordinates of the area covering the multi-route.

### getRoutes

```
{GeoObjectCollection} getRoutes()
```

**Returns** child collection of individual routes on the multi-route.

### getViaPoints

```
{GeoObjectCollection} getViaPoints()
```

**Returns** child collection of throughpoints for the multi-route.

### getWayPoints

```
{GeoObjectCollection} getWayPoints()
```

**Returns** child collection of waypoints for the multi-route.

### setActiveRoute

```
{ } setActiveRoute(route)
```

Sets the active route. The previous active route is deactivated and the [multiRouter.MultiRoute.event:activeroutechange](#) event is generated.

**Parameters:**

Parameter	Default value	Description
<code>route *</code>	—	Type: <code>multiRouter.driving.Route</code>   <code>multiRouter.masstransit.Route</code> null  Route to activate.

\* Mandatory parameter/option.

## multiRouter.MultiRouteModel

Extends [IEventEmitter](#).

The data model of a multi-stop route.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
multiRouter.MultiRouteModel(referencePoints[, params])
```

Creates the data model of a multi-stop route.

#### Parameters:

Parameter	Default value	Description
<code>referencePoints *</code>	—	Type: <a href="#">IMultiRouteReferencePoint</a> []  The description of the reference points on the multi-stop route.
<code>params</code>	—	Type: <a href="#">IMultiRouteParams</a>  Routing options.

\* Mandatory parameter/option.

### Example:

```
// Creating a model of the multi-stop route.
var multiRouteModel = new ymaps.multiRouter.MultiRouteModel(['Moscow', 'Tver', 'Peterburg'], {
  avoidTrafficJams: true,
  viaIndexes: [1]
});

// Creating a representation of a multi-stop route based on the model.
var multiRouteView = new ymaps.multiRouter.MultiRoute(multiRouteModel);
myMap.geoObjects.add(multiRouteView);

// Subscribing to the events of the multi-stop route model.
multiRouteView.model.events
  .add("requestsuccess", function (event) {
    var routes = event.get("target").getRoutes();
    console.log("Found routes: " + routes.length);
    for (var i = 0, l = routes.length; i < l; i++) {
      console.log("Route length " + (i + 1) + ": " + routes[i].properties.get("distance").text);
    }
  })
  .add("requestfail", function (event) {
    console.log("Error: " + event.get("error").message);
  });
```

### Fields

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

Name	Type	Description
<a href="#">properties</a>	<a href="#">data.Manager</a>	Multi-stop route data.

## Events

Name	Description
<a href="#">requestcancel</a>	<p>The data request has been canceled. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>referencePoints: Object[] - Array describing the set of reference points.</li> <li>params: Object - Routing options.</li> </ul>
<a href="#">requestchange</a>	<p>The reference data model of a multi-stop route (anchor points or routing options) has been changed. The result is a new data request to the routing service. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>referencePoints: Object[] - Array describing the set of reference points.</li> <li>params: Object - Routing parameters.</li> <li>oldReferencePoints: Object[] - Array describing the previous array of reference points.</li> <li>oldParams: Object - Previous routing options.</li> </ul>
<a href="#">requestfail</a>	<p>The request for data has failed. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>error: Error - error object.</li> </ul>
<a href="#">requestsend</a>	<p>A new request for the multi-stop route data model has been sent. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>referencePoints: Object[] - Array describing the set of reference points.</li> <li>params: Object - Routing options.</li> </ul>
<a href="#">requestsuccess</a>	<p>The request for data has completed successfully and the data model has been updated. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>init: Boolean - Indicates an initialization request.</li> <li>rough: Boolean - Flag for whether it is an intermediate request (used to optimize the size of the server response when editing a route).</li> <li>wayPointsChange: Boolean - Flag for whether the set of waypoints is changed.</li> <li>viaPointsChange: Boolean - Flag for whether the set of throughpoints is changed.</li> <li>routesChange: Boolean - Flag for whether the set of routes is changed.</li> </ul>

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destroys a model.
<a href="#">getAllPoints()</a>	( <a href="#">multiRouter.WayPointModel</a>   <a href="#">multiRouter.ViaPointModel</a> )[]	Returns the <a href="#">WayPointModel</a> array of models of throughpoints and waypoints, in the same order as the corresponding reference points.
<a href="#">getJSON()</a>	Object	Returns JSON data for the multi-stop route model.

Name	Returns	Description
<a href="#">getParams()</a>	<a href="#">IMultiRouteParams</a>	Returns the object with the current values of routing options.
<a href="#">getPoints()</a>	<a href="#">(multiRouter.WayPointModel   multiRouter.ViaPointModel)[]</a>	<del>Deprecated data field of the</del> multiRouter.MultiRouteModel.getAllPoints method. Not recommended for use.
<a href="#">getReferencePointIndexes()</a>	Object	Returns an object containing the following data fields: <ul style="list-style-type: none"> <li>way: Integer[] - Indexes of reference points corresponding to the set of the model's waypoints.</li> <li>via: Integer[] - Indexes of reference points corresponding to the set of the model's throughpoints.</li> </ul>
<a href="#">getReferencePoints()</a>	<a href="#">IMultiRouteReferencePoint[]</a>	Returns array of reference points on the multi-stop route.
<a href="#">getRoutes()</a>	<a href="#">multiRouter.driving.RouteModel[]</a>	Returns an array of models for child routes.
<a href="#">getViaPoints()</a>	<a href="#">multiRouter.ViaPointModel[]</a>	Returns an array of models for child throughpoints.
<a href="#">getWayPoints()</a>	<a href="#">multiRouter.WayPointModel[]</a>	Returns an array of models for child waypoints.
<a href="#">setParams(params[, extend[, clearRequests]])</a>		Sets routing options.
<a href="#">setReferencePoints(referencePoints[, viaIndexes[, clearRequests]])</a>		Sets the reference points on the multi-stop route.

## Fields details

### properties

```
{data.Manager} properties
```

Multi-stop route data.

### Events details

#### requestcancel

The data request has been canceled. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- referencePoints: Object[] - Array describing the set of reference points.
- params: Object - Routing options.

### requestchange

The reference data model of a multi-stop route (anchor points or routing options) has been changed. The result is a new data request to the routing service. Instance of the `Event` class. Names of fields that are available via the `Event.get` method:

- `referencePoints`: `Object[]` - Array describing the set of reference points.
- `params`: `Object` - Routing parameters.
- `oldReferencePoints`: `Object[]` - Array describing the previous array of reference points.
- `oldParams`: `Object` - Previous routing options.

### requestfail

The request for data has failed. Instance of the `Event` class. Names of fields that are available via the `Event.get` method:

- `error`: `Error` - error object.

### requestsend

A new request for the multi-stop route data model has been sent. Instance of the `Event` class. Names of fields that are available via the `Event.get` method:

- `referencePoints`: `Object[]` - Array describing the set of reference points.
- `params`: `Object` - Routing options.

### requestsuccess

The request for data has completed successfully and the data model has been updated. Instance of the `Event` class. Names of fields that are available via the `Event.get` method:

- `init`: `Boolean` - Indicates an initialization request.
- `rough`: `Boolean` - Flag for whether it is an intermediate request (used to optimize the size of the server response when editing a route).
- `wayPointsChange`: `Boolean` - Flag for whether the set of waypoints is changed.
- `viaPointsChange`: `Boolean` - Flag for whether the set of throughpoints is changed.
- `routesChange`: `Boolean` - Flag for whether the set of routes is changed.

## Methods details

### destroy

```
{ } destroy()
```

Destroys a model.

### getAllPoints

```
{ (multiRouter.WayPointModel|multiRouter.ViaPointModel) [] } getAllPoints()
```

**Returns** the combined array of models of throughpoints and waypoints, in the same order as the corresponding reference points.

### getJson

```
{Object} getJson()
```

**Returns** JSON data for the multi-stop route model.

### getParams

```
{IMultiRouteParams} getParams()
```

**Returns** the object with the current values of routing options.

### getPoints

```
{(multiRouter.WayPointModel|multiRouter.ViaPointModel)[]} getPoints()
```

Deprecated name of the `multiRouter.MultiRouteModel.getAllPoints` method. Not recommended for use.

**Returns** the combined array of models of throughpoints and waypoints, in the same order as the corresponding reference points.

### getReferencePointIndexes

```
{Object} getReferencePointIndexes()
```

**Returns** an object containing the following data fields:

- `way`: `Integer[]` - Indexes of reference points corresponding to the set of the model's waypoints.
- `via`: `Integer[]` - Indexes of reference points corresponding to the set of the model's throughpoints.

### getReferencePoints

```
{IMultiRouteReferencePoint[]} getReferencePoints()
```

**Returns** array of reference points on the multi-stop route.

### getRoutes

```
{multiRouter.driving.RouteModel[]|multiRouter.masstransit.RouteModel[]} getRoutes()
```

**Returns** an array of models for child routes.

### getViaPoints

```
{multiRouter.ViaPointModel[]} getViaPoints()
```

**Returns** an array of models for child throughpoints.

### getWayPoints

```
{multiRouter.WayPointModel[]} getWayPoints()
```

**Returns** an array of models for child waypoints.

### setParams

```
{ } setParams(params[, extend[, clearRequests]])
```

Sets routing options.

#### Parameters:

Parameter	Default value	Description
<code>params</code> *	—	Type: <code>IMultiRouteParams</code> Routing options.

Parameter	Default value	Description
<a href="#">extend</a>	false	Type: Boolean  Allows you to change only the specified set of parameters keeping all the rest of the values the same.
<a href="#">clearRequests</a>	false	Type: Boolean  Allows you to clear the queue of previous requests to the server.

\* Mandatory parameter/option.

### setReferencePoints

```
{ } setReferencePoints(referencePoints[, viaIndexes[, clearRequests]])
```

Sets the reference points on the multi-stop route.

#### Parameters:

Parameter	Default value	Description
<a href="#">referencePoints</a> *	—	Type: <a href="#">IMultiRouteReferencePoint[]</a>  Array of reference points.
<a href="#">viaIndexes</a>	—	Type: <a href="#">Integer[]</a>  Indexes of throughpoints in the array of reference points.
<a href="#">clearRequests</a>	false	Type: Boolean  Allows you to clear the queue of previous requests to the server.

\* Mandatory parameter/option.

## multiRouter.pedestrian

### multiRouter.pedestrian.Path

**Note:** The constructor of the `multiRouter.pedestrian.Path` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeoObject](#).

Representation of the pedestrian route path. A single route can contain several paths, and each path connects two waypoints.

[Fields](#) | [Events](#) | [Methods](#)

Creates a representation of the pedestrian route path.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">model</a>	<a href="#">multiRouter.pedestrian.PathModel</a>	Data model for the multiroute path.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">data.Manager</a>	Multi-stop route's path data. The following fields are available: <ul style="list-style-type: none"> <li>index: Integer - The sequential number of the path in the multi-stop route's corresponding route.</li> <li>type: String - The route type identifier; takes the value "pedestrian" for pedestrian routes.</li> <li>distance: Object - An object with the "text" and "value" fields that describes the length of the path in meters.</li> <li>duration: Object - An object with the "text" and "value" fields that describes the travel time of the path in seconds.</li> <li>coordinates: Number[][] - Coordinates of all points on the path.</li> <li>encodedCoordinates: String - A string of base64-encoded coordinates for all points on the path.</li> </ul>
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

## Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometrychange</a>	Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li> </ul> Inherited from <a href="#">IGeoObject</a> .

Name	Description
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldMap</code> - Old map.</li><li>• <code>newMap</code> - New map.</li></ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li><code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li><code>oldParent</code> - Old parent.</li> <li><code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">update</a>	<p>Updating the path rendering. Instance of the <a href="#">Event</a> class.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	<p>Returns reference to the map.</p> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	<p>Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>

Name	Returns	Description
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay.  Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IChildOnMap</a> .
<a href="#">getSegmentMarkers()</a>	<a href="#">GeoObjectCollection</a>	Returns a child collection of segment markers.
<a href="#">getSegments()</a>	<a href="#">GeoObjectCollection</a>	Returns the child collection of segments that the path consists of.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .

## Fields details

### model

```
{multiRouter.pedestrian.PathModel} model
```

Data model for the multiroute path.

### properties

```
{data.Manager} properties
```

Multi-stop route's path data. The following fields are available:

- **index**: Integer - The sequential number of the path in the multi-stop route's corresponding route.
- **type**: String - The route type identifier; takes the value "pedestrian" for pedestrian routes.
- **distance**: Object - An object with the "text" and "value" fields that describes the length of the path in meters.
- **duration**: Object - An object with the "text" and "value" fields that describes the travel time of the path in seconds.
- **coordinates**: Number[][] - Coordinates of all points on the path.
- **encodedCoordinates**: String - A string of base64-encoded coordinates for all points on the path.

## Events details

### update

Updating the path rendering. Instance of the [Event](#) class.

## Methods details

### getSegmentMarkers

```
{GeoObjectCollection} getSegmentMarkers()
```

**Returns** a child collection of segment markers.

## getSegments

```
{GeoObjectCollection} getSegments()
```

**Returns** the child collection of segments that the path consists of.

### multiRouter.pedestrian.PathModel

**Note:** The constructor of the `multiRouter.pedestrian.PathModel` class is hidden, as this class is not intended for autonomous initialization.

Extends [IEventEmitter](#).

A data model of the pedestrian route. A single route can contain several paths, and each path connects two waypoints.

[Fields](#) | [Events](#) | [Methods](#)

Creates a data model of the pedestrian route.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">properties</a>	<a href="#">data.Manager</a>	Multi-stop route's path data. The following fields are available: <ul style="list-style-type: none"> <li><code>index</code>: Integer - The sequential number of the path in the multi-stop route's corresponding route.</li> <li><code>type</code>: String - The route type identifier; takes the value "pedestrian" for pedestrian routes.</li> <li><code>distance</code>: Object - An object with the "text" and "value" fields that describes the length of the path in meters.</li> <li><code>duration</code>: Object - An object with the "text" and "value" fields that describes the travel time of the path in seconds.</li> <li><code>coordinates</code>: Number[][] - Coordinates of all points on the path.</li> <li><code>encodedCoordinates</code>: String - A string of base64-encoded coordinates for all points on the path.</li> </ul>
<a href="#">route</a>	<a href="#">multiRouter.pedestrian.RouteModel</a>	Reference to the parent model of the route.

### Events

Name	Description
<a href="#">update</a>	Updating the model with new data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>segmentsChange</code>: Boolean - Flag for whether the set of segments is changed</li> </ul>

### Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destroys a model.
<a href="#">getSegments()</a>	<a href="#">multiRouter.pedestrian.SegmentCollection</a>	Returns array of path segments.

Name	Returns	Description
<code>getType()</code>	String	Returns ID of the route type. For pedestrian routes, returns the string "pedestrian".
<code>update(pathJson)</code>		Updates the state of the model.

## Fields details

### properties

```
{data.Manager} properties
```

Multi-stop route's path data. The following fields are available:

- `index`: Integer - The sequential number of the path in the multi-stop route's corresponding route.
- `type`: String - The route type identifier; takes the value "pedestrian" for pedestrian routes.
- `distance`: Object - An object with the "text" and "value" fields that describes the length of the path in meters.
- `duration`: Object - An object with the "text" and "value" fields that describes the travel time of the path in seconds.
- `coordinates`: Number[][] - Coordinates of all points on the path.
- `encodedCoordinates`: String - A string of base64-encoded coordinates for all points on the path.

### route

```
{multiRouter.pedestrian.RouteModel} route
```

Reference to the parent model of the route.

### Events details

#### update

Updating the model with new data. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `segmentsChange`: Boolean - Flag for whether the set of segments is changed

### Methods details

#### destroy

```
{ } destroy()
```

Destroys a model.

#### getSegments

```
{multiRouter.pedestrian.SegmentModel[]} getSegments()
```

Returns array of path segments.

#### getType

```
{String} getType()
```

Returns ID of the route type. For pedestrian routes, returns the string "pedestrian".

**update**

```
{ } update (pathJson)
```

Updates the state of the model.

**Parameters:**

Parameter	Default value	Description
<a href="#">pathJson</a> *	—	Type: Object JSON data.

\* Mandatory parameter/option.

**multiRouter.pedestrian.Route**

**Note:** The constructor of the `multiRouter.pedestrian.Route` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeoObject](#).

Displaying an individual pedestrian route. A multi-stop route can consist of several individual routes.

[Fields](#) | [Events](#) | [Methods](#)

Creates the representation to display an individual pedestrian route.

**Fields**

Name	Type	Description
<a href="#">balloon</a>	<a href="#">IMultiRouterRouteBalloon</a>	Balloon of a route.
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">model</a>	<a href="#">multiRouter.pedestrian.RouteModel</a>	The data model of an individual route.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">data.Manager</a>	The route data. The following fields are available: <ul style="list-style-type: none"> <li><code>index</code>: Integer - The ordinal number of the route in a multi-stop route.</li> <li><code>type</code>: String - The route type identifier; takes the value "pedestrian" for pedestrian routes.</li> <li><code>distance</code>: Object - An object with the "text" and "value" fields that specifies the length of the route in meters.</li> <li><code>duration</code>: Object - An object with the "text" and "value" fields that describes the travel time of the route in seconds.</li> <li><code>boundedBy</code>: Number[][] - Coordinates of the upper and lower corners of the rectangle that bounds the route.</li> </ul>
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

## Events

Name	Description
<a href="#">balloonclose</a>	Closing the balloon.
<a href="#">balloonopen</a>	Opening the balloon.
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li></ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>oldMap - Old map.</li><li>newMap - New map.</li></ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">update</a>	<p>Updating the route rendering. Instance of the <a href="#">Event</a> class.</p>

Name	Description
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	<p>Returns reference to the map.</p> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	<p>Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	<p>The method provides synchronous access to the overlay.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	<p>Returns link to the parent object, or null if the parent element was not set.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>
<a href="#">getPaths()</a>	<a href="#">GeoObjectCollection</a>	<p>Returns a child collection of paths that make up the route.</p>
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	<p>Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>

## Fields details

### balloon

```
{IMultiRouterRouteBalloon} balloon
```

Balloon of a route.

### model

```
{multiRouter.pedestrian.RouteModel} model
```

The data model of an individual route.

### properties

```
{data.Manager} properties
```

The route data. The following fields are available:

- **index**: Integer - The ordinal number of the route in a multi-stop route.
- **type**: String - The route type identifier; takes the value "pedestrian" for pedestrian routes.
- **distance**: Object - An object with the "text" and "value" fields that specifies the length of the route in meters.
- **duration**: Object - An object with the "text" and "value" fields that describes the travel time of the route in seconds.
- **boundedBy**: Number[][] - Coordinates of the upper and lower corners of the rectangle that bounds the route.

### Events details

#### balloonclose

Closing the balloon.

#### balloonopen

Opening the balloon.

#### update

Updating the route rendering. Instance of the [Event](#) class.

### Methods details

#### getPaths

```
{GeoObjectCollection} getPaths()
```

**Returns** a child collection of paths that make up the route.

#### multiRouter.pedestrian.RouteModel

**Note:** The constructor of the multiRouter.pedestrian.RouteModel class is hidden, as this class is not intended for autonomous initialization.

Extends [IEventEmitter](#).

A data model of a single pedestrian route. A multi-stop route can consist of several individual routes.

[Fields](#) | [Events](#) | [Methods](#)

Creates a data model of a single pedestrian route.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">multiRoute</a>	<a href="#">multiRouter.MultiRouteModel</a>	Reference to the parent model of a multi-stop route.
<a href="#">properties</a>	<a href="#">data.Manager</a>	The route data. The following fields are available: <ul style="list-style-type: none"> <li>• <b>index</b>: Integer - The ordinal number of the route in a multi-stop route.</li> <li>• <b>type</b>: String - The route type identifier; takes the value "pedestrian" for pedestrian routes.</li> <li>• <b>distance</b>: Object - An object with the "text" and "value" fields that specifies the length of the route in meters.</li> <li>• <b>duration</b>: Object - An object with the "text" and "value" fields that describes the travel time of the route in seconds.</li> <li>• <b>boundedBy</b>: Number[][] - Coordinates of the upper and lower corners of the rectangle that bounds the route.</li> </ul>

## Events

Name	Description
<a href="#">update</a>	Updating the model with new data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>pathsChange</code>: Boolean - Flag for whether the set of paths is changed.</li> </ul>

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destroys a model.
<a href="#">getPaths()</a>	<a href="#">multiRouter.pedestrian.PathModel</a>	Returns array of route paths.
<a href="#">getType()</a>	String	Returns ID of the route type. For pedestrian routes, returns the string "pedestrian".
<a href="#">update(routeJson)</a>		Updates the state of the model.

## Fields details

### multiRoute

```
{multiRouter.MultiRouteModel} multiRoute
```

Reference to the parent model of a multi-stop route.

### properties

```
{data.Manager} properties
```

The route data. The following fields are available:

- `index`: Integer - The ordinal number of the route in a multi-stop route.
- `type`: String - The route type identifier; takes the value "pedestrian" for pedestrian routes.
- `distance`: Object - An object with the "text" and "value" fields that specifies the length of the route in meters.
- `duration`: Object - An object with the "text" and "value" fields that describes the travel time of the route in seconds.
- `boundedBy`: Number[][] - Coordinates of the upper and lower corners of the rectangle that bounds the route.

## Events details

### update

Updating the model with new data. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `pathsChange`: Boolean - Flag for whether the set of paths is changed.

## Methods details

### destroy

```
{ } destroy()
```

Destroys a model.

## getPaths

```
{multiRouter.pedestrian.PathModel[]} getPaths()
```

**Returns** array of route paths.

## getType

```
{String} getType()
```

**Returns** ID of the route type. For pedestrian routes, returns the string "pedestrian".

## update

```
{ } update(routeJson)
```

Updates the state of the model.

### Parameters:

Parameter	Default value	Description
<code>routeJson</code> *	—	Type: Object JSON data.

\* Mandatory parameter/option.

## multiRouter.pedestrian.SegmentModel

**Note:** The constructor of the `multiRouter.pedestrian.SegmentModel` class is hidden, as this class is not intended for autonomous initialization.

Extends [IEventEmitter](#).

A data model of a pedestrian route segment.

[Fields](#) | [Events](#) | [Methods](#)

Creates a data model of a pedestrian route segment.

### Fields

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<code>geometry</code>	<a href="#">geometry.base.LineString</a>	Geometry of a segment.
<code>path</code>	<a href="#">multiRouter.pedestrian.PathModel</a>	Reference to the parent model of the path.

Name	Type	Description
<a href="#">properties</a>	<a href="#">data.Manager</a>	<p>Segment data. The following fields are available:</p> <ul style="list-style-type: none"> <li>index: Integer - The ordinal number of the segment in the array of segments of the corresponding route path.</li> <li>type: String - Segment type identifier, which takes the value "pedestrian" for pedestrian segments.</li> <li>text: String - Text description of the segment.</li> <li>distance: Object - An object with the "text" and "value" fields that specifies the length of the segment in meters.</li> <li>duration: Object - An object with the "text" and "value" fields that describes the travel time of the segment in seconds.</li> <li>lodIndex: Integer - The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.</li> </ul>

## Events

Name	Description
<a href="#">update</a>	Updating the model with new data. Instance of the <a href="#">Event</a> class.

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destroys a model.
<a href="#">getType()</a>	String	Returns ID of the segment type. For walking segments of pedestrian routes, returns the string "pedestrian".

## Fields details

### geometry

```
{geometry.base.LineString} geometry
```

Geometry of a segment.

### path

```
{multiRouter.pedestrian.PathModel} path
```

Reference to the parent model of the path.

### properties

```
{data.Manager} properties
```

Segment data. The following fields are available:

- index: Integer - The ordinal number of the segment in the array of segments of the corresponding route path.
- type: String - Segment type identifier, which takes the value "pedestrian" for pedestrian segments.
- text: String - Text description of the segment.
- distance: Object - An object with the "text" and "value" fields that specifies the length of the segment in meters.
- duration: Object - An object with the "text" and "value" fields that describes the travel time of the segment in seconds.

- `lodIndex`: Integer - The ordinal number of the first throughpoint of the segment in the full set of coordinates of the corresponding route path.

### Events details

#### update

Updating the model with new data. Instance of the [Event](#) class.

### Methods details

#### destroy

```
{ } destroy()
```

Destroys a model.

#### getType

```
{String} getType()
```

**Returns** ID of the segment type. For walking segments of pedestrian routes, returns the string "pedestrian".

### multiRouter.pedestrian.WalkSegment

**Note:** The constructor of the `multiRouter.pedestrian.WalkSegment` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeoObject](#).

Representation of a segment on a pedestrian route.

[Fields](#) | [Events](#) | [Methods](#)

Creates the representation to display a segment on a pedestrian route.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">IDataManager</a>	Geo object data. Inherited from <a href="#">IGeoObject</a> .
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

## Events

Name	Description
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li></ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>oldMap - Old map.</li><li>newMap - New map.</li></ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"><li>• <code>overlay</code>: <code>IOverlay</code> null - Reference to the overlay.</li><li>• <code>oldOverlay</code>: <code>IOverlay</code> null - Previous overlay of the geo object.</li></ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldParent</code> - Old parent.</li><li>• <code>newParent</code> - New parent.</li></ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"><li>• <code>originalEvent</code>: <code>IEvent</code> - Original event of the data manager.</li></ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IChildOnMap</a> .

**multiRouter.ViaPoint**

**Note:** The constructor of the `multiRouter.ViaPoint` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeoObject](#).

Representation of a throughpoint. Throughpoints do not mean a stop. Thus, when passing via a throughpoint, the segment of the route path doesn't break.

[Fields](#) | [Events](#) | [Methods](#)

Creates a representation for displaying the throughpoint.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">model</a>	<a href="#">multiRouter.ViaPointModel</a>	The data model of a throughpoint.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

Name	Type	Description
<a href="#">properties</a>	<a href="#">data.Manager</a>	Data for a multi-stop route throughpoint. The following fields are available: <ul style="list-style-type: none"> <li>index: Integer - The sequential number of the point.</li> <li>lodIndex: Integer - The ordinal number of the throughpoint in the full set of coordinates of the corresponding route path.</li> </ul>
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

## Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometrychange</a>	Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li> </ul> Inherited from <a href="#">IGeoObject</a> .
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> Inherited from <a href="#">IParentOnMap</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseleave</a>	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>

Name	Description
<a href="#">propertieschange</a>	Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> Inherited from <a href="#">IGeoObject</a> .
<a href="#">update</a>	Updating a representation for displaying the throughpoint. Instance of the <a href="#">Event</a> class.
<a href="#">wheel</a>	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay. <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. <p>Inherited from <a href="#">IChildOnMap</a>.</p>
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. <p>Inherited from <a href="#">IChildOnMap</a>.</p>

## Fields details

### model

```
{multiRouter.ViaPointModel} model
```

The data model of a throughpoint.

### properties

```
{data.Manager} properties
```

Data for a multi-stop route throughpoint. The following fields are available:

- `index`: Integer - The sequential number of the point.
- `lodIndex`: Integer - The ordinal number of the throughpoint in the full set of coordinates of the corresponding route path.

### Events details

#### update

Updating a representation for displaying the throughpoint. Instance of the [Event](#) class.

## multiRouter.ViaPointModel

**Note:** The constructor of the `multiRouter.ViaPointModel` class is hidden, as this class is not intended for autonomous initialization.

Extends [IEventManager](#).

The data model of a throughpoint on the multi-stop route. Throughpoints do not signify a stop. When passing via a throughpoint, the segment of the route path doesn't break.

[Fields](#) | [Events](#) | [Methods](#)

Creates the data model of a multi-stop route throughpoint.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">geometry.base.Point</a>	Geometry of a multi-stop route throughpoint.
<a href="#">multiRoute</a>	<a href="#">multiRouter.MultiRouteModel</a>	Reference to the parent model of a multi-stop route.
<a href="#">properties</a>	<a href="#">data.Manager</a>	Data for a multi-stop route throughpoint. The following fields are available: <ul style="list-style-type: none"> <li>• <code>index</code>: Integer - The sequential number of the point.</li> <li>• <code>lodIndex</code>: Integer - The ordinal number of the throughpoint in the full set of coordinates of the corresponding route path.</li> </ul>

### Events

Name	Description
<a href="#">referencepointchange</a>	Change to the reference point. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>• <code>oldReferencePoint</code>: Object - Description of the previous reference point.</li> </ul>
<a href="#">update</a>	Updating the model with new data. Instance of the <a href="#">Event</a> class.

### Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destroys a model.
<a href="#">getReferencePoint()</a>	Object	Returns the corresponding reference point.

Name	Returns	Description
<a href="#">getReferencePointIndex()</a>	Integer	Returns the index corresponding to a reference point in the set of reference points for the parent multi-stop route.
<a href="#">setReferencePoint(referencePoint)</a>		Sets the description for the corresponding reference point. A reference point can be set using one of the following ways: <ul style="list-style-type: none"> <li>• A string containing the postal address of the reference point.</li> <li>• An array containing the latitude and longitude of the reference point.</li> <li>• A <code>geometry.Point</code> geometry describing the reference point.</li> </ul>
<a href="#">update(viaPointJson)</a>		Updates the model with new data.

### Fields details

#### geometry

```
{geometry.base.Point} geometry
```

Geometry of a multi-stop route throughpoint.

#### multiRoute

```
{multiRouter.MultiRouteModel} multiRoute
```

Reference to the parent model of a multi-stop route.

#### properties

```
{data.Manager} properties
```

Data for a multi-stop route throughpoint. The following fields are available:

- `index`: Integer - The sequential number of the point.
- `lodIndex`: Integer - The ordinal number of the throughpoint in the full set of coordinates of the corresponding route path.

### Events details

#### referencepointchange

Change to the reference point. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `oldReferencePoint`: Object - Description of the previous reference point.

#### update

Updating the model with new data. Instance of the [Event](#) class.

## Methods details

### destroy

```
{ } destroy()
```

Destroys a model.

### getReferencePoint

```
{Object} getReferencePoint()
```

**Returns** the corresponding reference point.

### getReferencePointIndex

```
{Integer} getReferencePointIndex()
```

**Returns** the index corresponding to a reference point in the set of reference points for the parent multi-stop route.

### setReferencePoint

```
{ } setReferencePoint(referencePoint)
```

Sets the description for the corresponding reference point. A reference point can be set using one of the following ways:

- A string containing the postal address of the reference point.
- An array containing the latitude and longitude of the reference point.
- A `geometry.Point` geometry describing the reference point.

#### Parameters:

Parameter	Default value	Description
<code>referencePoint</code> *	—	Type: Object  Description of the reference point.

\* Mandatory parameter/option.

### update

```
{ } update(viaPointJson)
```

Updates the model with new data.

#### Parameters:

Parameter	Default value	Description
<code>viaPointJson</code> *	—	Type: Object  JSON data.

\* Mandatory parameter/option.

## multiRouter.WayPoint

**Note:** The constructor of the `multiRouter.WayPoint` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeoObject](#).

Representation of a waypoint. A waypoint means a stop, and waypoints divide the route into so-called paths.

[Fields](#) | [Events](#) | [Methods](#)

Creates a representation for displaying the waypoint.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">model</a>	<a href="#">multiRouter.WayPointModel</a>	The data model of a waypoint.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">data.Manager</a>	Data for a multi-stop route waypoint. The following fields are available: <ul style="list-style-type: none"> <li>index: Integer - The sequential number of the point.</li> <li>request: String - The request corresponding to the point of the request.</li> <li>address: String - The postal address of the point.</li> <li>description: String - Description of the point.</li> <li>name: String - Name of the point.</li> <li>geocoderMetaData: Object - Geocoder metadata.</li> </ul>
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

## Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometrychange</a>	Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li> </ul> Inherited from <a href="#">IGeoObject</a> .

Name	Description
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldMap</code> - Old map.</li><li>• <code>newMap</code> - New map.</li></ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li><code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li><code>oldParent</code> - Old parent.</li> <li><code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li><code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">update</a>	<p>Updating a representation for displaying the waypoint. Instance of the <a href="#">Event</a> class.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	<p>Returns reference to the map.</p> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	<p>Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>

Name	Returns	Description
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay.  Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IChildOnMap</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .

## Fields details

### model

```
{multiRouter.WayPointModel} model
```

The data model of a waypoint.

### properties

```
{data.Manager} properties
```

Data for a multi-stop route waypoint. The following fields are available:

- `index`: Integer - The sequential number of the point.
- `request`: String - The request corresponding to the point of the request.
- `address`: String - The postal address of the point.
- `description`: String - Description of the point.
- `name`: String - Name of the point.
- `geocoderMetaData`: Object - Geocoder metadata.

## Events details

### update

Updating a representation for displaying the waypoint. Instance of the [Event](#) class.

## multiRouter.WayPointModel

**Note:** The constructor of the `multiRouter.WayPointModel` class is hidden, as this class is not intended for autonomous initialization.

Extends [IEventEmitter](#).

The data model of a multi-stop route waypoint. A waypoint means a stop, and waypoints divide the route into so-called paths.

[Fields](#) | [Events](#) | [Methods](#)

Creates the data model of a multi-stop route waypoint.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">geometry.base.Point</a>	Geometry of a multi-stop route waypoint.
<a href="#">multiRoute</a>	<a href="#">multiRouter.MultiRouteModel</a>	Reference to the parent model of a multi-stop route.
<a href="#">properties</a>	<a href="#">data.Manager</a>	Data for a multi-stop route waypoint. The following fields are available: <ul style="list-style-type: none"> <li>index: Integer - The sequential number of the point.</li> <li>request: String - The request corresponding to the point of the request.</li> <li>address: String - The postal address of the point.</li> <li>description: String - Description of the point.</li> <li>name: String - Name of the point.</li> <li>geocoderMetaData: Object - Geocoder metadata.</li> </ul>

**Events**

Name	Description
<a href="#">referencepointchange</a>	Change to the reference point. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>oldReferencePoint: Object - Description of the previous reference point.</li> </ul>
<a href="#">update</a>	Updating the model with new data. Instance of the <a href="#">Event</a> class.

**Methods**

Name	Returns	Description
<a href="#">destroy()</a>		Destroys a model.
<a href="#">getReferencePoint()</a>	Object	Returns the corresponding reference point.
<a href="#">getReferencePointIndex()</a>	Integer	Returns the index corresponding to a reference point in the set of reference points for the parent multi-stop route.
<a href="#">setReferencePoint(referencePoint)</a>		Sets the description for the corresponding reference point. A reference point can be set using one of the following ways: <ul style="list-style-type: none"> <li>A string containing the postal address of the reference point.</li> <li>An array containing the latitude and longitude of the reference point.</li> <li>A <a href="#">geometry.Point</a> geometry describing the reference point.</li> </ul>

Name	Returns	Description
<code>update(wayPointJson)</code>		Updates the model with new data.

## Fields details

### geometry

```
{geometry.base.Point} geometry
```

Geometry of a multi-stop route waypoint.

### multiRoute

```
{multiRouter.MultiRouteModel} multiRoute
```

Reference to the parent model of a multi-stop route.

### properties

```
{data.Manager} properties
```

Data for a multi-stop route waypoint. The following fields are available:

- `index`: Integer - The sequential number of the point.
- `request`: String - The request corresponding to the point of the request.
- `address`: String - The postal address of the point.
- `description`: String - Description of the point.
- `name`: String - Name of the point.
- `geocoderMetaData`: Object - Geocoder metadata.

## Events details

### referencepointchange

Change to the reference point. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `oldReferencePoint`: Object - Description of the previous reference point.

### update

Updating the model with new data. Instance of the [Event](#) class.

## Methods details

### destroy

```
{ } destroy()
```

Destroys a model.

### getReferencePoint

```
{Object} getReferencePoint()
```

**Returns** the corresponding reference point.

### getReferencePointIndex

```
{Integer} getReferencePointIndex()
```

**Returns** the index corresponding to a reference point in the set of reference points for the parent multi-stop route.

### setReferencePoint

```
{ } setReferencePoint(referencePoint)
```

Sets the description for the corresponding reference point. A reference point can be set using one of the following ways:

- A string containing the postal address of the reference point.
- An array containing the latitude and longitude of the reference point.
- A `geometry.Point` geometry describing the reference point.

#### Parameters:

Parameter	Default value	Description
<code>referencePoint</code> *	—	Type: Object  Description of the reference point.

\* Mandatory parameter/option.

### update

```
{ } update(wayPointJson)
```

Updates the model with new data.

#### Parameters:

Parameter	Default value	Description
<code>wayPointJson</code> *	—	Type: Object  JSON data.

\* Mandatory parameter/option.

## ObjectManager

Extends [ICustomizable](#), [IEventEmitter](#), [IGeoObject](#), [IParentOnMap](#).

Object manager. Allows optimally displaying, clustering, and managing visibility for objects. Note that objects drawn on the map via this manager can't have editing and dragging modes enabled.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
ObjectManager([options])
```

#### Parameters:

Parameter	Default value	Description
<a href="#">options</a>	—	<p>Type: Object</p> <p>Options.</p> <ul style="list-style-type: none"> <li>You can set all the options specified in the <a href="#">Clusterer</a> description, except for <code>hasBalloon</code> and <code>hasHint</code> options.</li> <li>Cluster options are set with the "cluster" prefix. The list of options is specified in the description of <a href="#">ClusterPlacemark</a>;</li> <li>Options for singular objects should be specified with the <code>geoObject</code> prefix. The list of options is specified in <a href="#">GeoObject</a>. Note that the manager ignores the 'visible' option.</li> </ul>
<a href="#">options.clusterize</a>	false	<p>Type: Boolean</p> <p>Flag indicating whether the objects should be clusterized. Note that clusterization only works for point objects at this time. When cluster mode is enabled, all non-point objects are ignored.</p>
<a href="#">options.syncOverlayInit</a>	false	<p>Type: Boolean</p> <p>A flag that allows creating overlays for objects synchronously. Note that when you create an overlay synchronously, you should ensure that the appropriate class, which implements the <code>IOverlay</code> interface, is loaded. By default, the overlays are created asynchronously, and the overlay class is loaded on demand.</p>
<a href="#">options.viewportMargin</a>	128	<p>Type: Number Number[]</p> <p>Offset for the area where the objects are shown. Use this option to expand the view area of objects relative to the visible area of the map.</p>

**Example:**

```

var objectManager = new ymaps.ObjectManager({
  // Enabling clustering.
  clusterize: true,
  // Cluster options are set with the "cluster" prefix.
  clusterHasBalloon: false,
  // Geo object options are set with the "geoObject" prefix.
  geoObjectOpenBalloonOnClick: false
});

// You can set options directly for child collections.
objectManager.clusters.options.set({
  preset: 'islands#grayClusterIcons',
  hintContentLayout: ymaps.templateLayoutFactory.createClass('Group of objects')
});
objectManager.objects.options.set('preset', 'islands#grayIcon');

```

**Fields**

Name	Type	Description
<a href="#">clusters</a>	<a href="#">objectManager.ClusterCollection</a>	Collection of clusters generated by the manager.
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">objects</a>	<a href="#">objectManager.ObjectCollection</a>	Collection of objects added to the layer.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">IDataManager</a>	Geo object data. Inherited from <a href="#">IGeoObject</a> .
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

**Events**

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometrychange</a>	Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li> </ul> Inherited from <a href="#">IGeoObject</a> .
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> Inherited from <a href="#">IParentOnMap</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>

Name	Description
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>overlay: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>oldOverlay: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">add(objects)</a>	<a href="#">ObjectManager</a>	Returns reference to the object manager.
<a href="#">getBounds()</a>	Number[][] null	Calculates the boundaries in geo coordinates for an area that covers all the objects in the manager.
<a href="#">getFilter()</a>	String Function null	Returns the set filter function.
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getObjectState(id)</a>	Object	Getting information about the current state of an object added to the manager.
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay. Inherited from <a href="#">IGeoObject</a> .

Name	Returns	Description
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IChildOnMap</a> .
<a href="#">getPixelBounds()</a>	Number[][] null	Calculates the boundaries in global pixel coordinates for an area that covers all the objects in the manager.
<a href="#">remove(objects)</a>	<a href="#">ObjectManager</a>	Returns reference to the object manager.
<a href="#">removeAll()</a>	<a href="#">ObjectManager</a>	Deleting all objects from the manager.
<a href="#">setFilter(filterFunction)</a>		Sets a filter function for objects.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .

## Fields details

### clusters

```
{objectManager.ClusterCollection} clusters
```

Collection of clusters generated by the manager.

#### Example:

```
objectManager.clusters.events.add('click', function (e) {
  var objectId = e.get('objectId');
  objectManager.clusters.balloon.open(objectId);
});
```

### objects

```
{objectManager.ObjectCollection} objects
```

Collection of objects added to the layer.

#### Example:

```
objectManager.objects.events.add('click', function (e) {
  var objectId = e.get('objectId');
  objectManager.objects.balloon.open(objectId);
});
```

## Methods details

### add

```
{ObjectManager} add(objects)
```

**Returns** reference to the object manager.

**Parameters:**

Parameter	Default value	Description
<code>objects</code> *	—	<p>Type: Object Object[] String</p> <p>String or object with a JSON description of the objects. JSON object descriptions are formed using the following approach (see the example below). An object may be an entity or a collection of entities. A collection of entities is made up of an object with the following fields:</p> <ul style="list-style-type: none"> <li><code>type</code> - Type of object. The value of the field must be "FeatureCollection".</li> <li><code>features</code> - Array of child entities in the collection. Child objects may be entities or nested collections of entities.</li> </ul> <p>An entity is made up of an object with the following fields:</p> <ul style="list-style-type: none"> <li><code>id</code> - Unique object ID. Mandatory field.</li> <li><code>type</code> - Type of object. The value of the field must be "Feature".</li> <li><code>geometry</code> - Object geometry. Contains the "type" and "coordinates" fields. The value corresponds to the <a href="#">GeoObject</a> passed to the constructor.</li> <li><code>options</code> - Options for the geo object.</li> <li><code>properties</code> - Geo object data.</li> </ul>

\* Mandatory parameter/option.

**Examples:****1.**

```
var objectManager = new ymaps.ObjectManager({ clusterize: true });
var currentId = 0;

// Adding a single object.
objectManager.add({
  type: 'Feature',
  id: currentId++,
  geometry: {
    type: 'Point',
    coordinates: [56.23, 34.79]
  },
  properties: {
    hintContent: 'Popup hint text',
    balloonContent: 'Balloon content'
  }
});
map.geoObjects.add(objectManager);
```

**2.**

```
// Adding an array of point objects.
var objects = [];
for (var i = 0, l = coordinates.length; i < l; i++) {
  objects.push({
    type: 'Feature',
    id: currentId++,
    geometry: {
      type: 'Point',
      coordinates: coordinates[i]
    }
  });
}
objectManager.add(objects);
map.geoObjects.add(objectManager);
```

**3.**

```
// Adding collections of objects.
var collection = {
  type: 'FeatureCollection',
  features: [{
    type: 'Feature',
    id: currentId++,
    geometry: {
      type: 'Point',
      coordinates: [24.34, 65.24]
    }
  }, {
    type: 'Feature',
    id: currentId++,
    geometry: {
      type: 'Point',
      coordinates: [25.34, 63.24]
    }
  }
]}
objectManager.add(collection);
map.geoObjects.add(objectManager);
```

**4.**

```
// Adding non-point objects
var objects = [];

// Adding a circle
objects.push({
  type: 'Feature',
  id: 1,
  geometry: {
    type: 'Circle',
    coordinates: [55.755381, 37.619044],
    radius: 300
  }
});

// Adding a rectangle
objects.push({
  type: 'Feature',
  id: 2,
  geometry: {
    type: 'Rectangle',
    coordinates: [
      [55.764286, 37.606169],
      [55.759688, 37.620588]
    ]
  },
  options: {
    fillColor: '#FFFFFF',
    opacity: 0.8
  }
});

// Adding a polyline
objects.push({
  type: 'Feature',
  id: 3,
  geometry: {
    type: 'LineString',
    coordinates: [
      [55.75901100, 37.6308886],
      [55.7516538, 37.6299444],
      [55.74603822, 37.6380125]
    ]
  },
  options: {
    strokeColor: "#FF0000",
    strokeWidth: 5
  }
});
```

```

});

// Adding a polygon
objects.push({
  type: 'Feature',
  id: 4,
  geometry: {
    type: 'Polygon',
    coordinates: [[
      [55.75175065, 37.6041094],
      [55.75005637, 37.6137224],
      [55.742891186, 37.6166407],
      [55.74153546, 37.60342281],
      [55.74700649, 37.59775798]
    ]]
  },
  options: {
    opacity: 0.2,
    strokeWidth: 2,
    fillColor: '#00FF00'
  }
});

objectManager.add(objects);

```

### getBounds

```
{Number[][]|null} getBounds()
```

Calculates the boundaries in geo coordinates for an area that covers all the objects in the manager.

**Returns** array of the area's coordinates, or null if the manager has not been added to the map.

### getFilter

```
{String|Function|null} getFilter()
```

**Returns** the set filter function.

### getObjectState

```
{Object} getObjectState(id)
```

Getting information about the current state of an object added to the manager.

**Returns** object with following fields:

- **found** - Attribute that indicates whether an object with the passed ID exists. Type: Boolean.
- **isShown** - Attribute that indicates whether the object is located in the visible area of the map. Type: Boolean.
- **cluster** - JSON description of the cluster the object was added to. Besides the required fields, it contains the `properties.geoObjects` field with an array of objects that are in the cluster. This field is returned only when clusterization is enabled.
- **isClustered** - Attribute indicating whether an object is in the cluster. This field is returned only when clusterization is enabled. Type: Boolean.
- **isFilteredOut** - Attribute indicating whether an object passed through filtration. If the filter is not set or the object passed through filtration, the value of the field is "false". Type: Boolean.

#### Parameters:

Parameter	Default value	Description
<code>id</code> *	—	Type: Object  ID of the object to get the state for.

\* Mandatory parameter/option.

**Example:**

```
// Opening the cluster balloon with the selected object.
// Getting data about the state of an object inside the cluster.
var objectState = objectManager.getObjectState(objects[1].id);
// Checking whether the object is located in the visible area of the map.
if (objectState.found && objectState.isShown) {
  // If the object is in a cluster, we open the cluster balloon with the appropriate object selected.
  if (objectState.isClustered) {
    objectManager.clusters.state.set('activeObject', objects[1]);
    objectManager.clusters.balloon.open(objectState.cluster.id);
  } else {
    // If the object isn't in a cluster, we open its own balloon.
    objectManager.objects.balloon.open(objects[i].id);
  }
}
}
```

**getPixelBounds**

```
{Number[][]|null} getPixelBounds()
```

Calculates the boundaries in global pixel coordinates for an area that covers all the objects in the manager.

**Returns** array of the area's coordinates, or null if the manager has not been added to the map.

**remove**

```
{ObjectManager} remove(objects)
```

**Returns** reference to the object manager.

**Parameters:**

Parameter	Default value	Description
<code>objects *</code>	—	Type: Object Object[] String  A string object with a JSON description of objects or an array of IDs of the objects being deleted. For the object description format, see the description of the method <a href="#">ObjectManager.add</a>

\* Mandatory parameter/option.

**Examples:****1.**

```
// Removing objects with IDs 0 and 56.
objectManager.remove([0, 56]);
```

**2.**

```
// Deleting an array of objects
objectManager.remove([
  {
    type: 'Feature',
    id: 23,
    geometry: {
      type: 'Point',
      coordinates: [45.33, 24.33]
    }
  },
  {
    type: 'Feature',
    id: 52,
    geometry: {
      type: 'Point',
      coordinates: [45.23, 24.34]
    }
  }
],
```

```
});
```

### 3.

```
// Deleting a collection of objects.
objectManager.remove({
  type: 'FeatureCollection',
  features: [
    // Describing objects in the collection
    ...
  ]
});
```

## removeAll

```
{ObjectManager} removeAll()
```

Deleting all objects from the manager.

**Returns** self-reference.

### Example:

```
objectManager.removeAll();
```

## setFilter

```
{ObjectManager} setFilter(filterFunction)
```

Sets a filter function for objects.

### Parameters:

Parameter	Default value	Description
<code>filterFunction</code> *	—	<p>Type: Function String</p> <p>Filter function. Takes a single object added to ObjectManager. If the function returns true, the object will be processed. If false, the object will be excluded from further processing. A string can also be passed as a filter. The following keywords are available in the string filter:</p> <ul style="list-style-type: none"> <li>options - Accessing the object's options.</li> <li>properties - Accessing the object's data.</li> <li>geometry - Accessing the object's geometry.</li> <li>id - Accessing the object's identifier.</li> </ul> <p>For a filter, you can specify an expression that returns true or false.</p>

\* Mandatory parameter/option.

### Examples:

1.

```
// Skipping objects with an ID under 100.
objectManager.setFilter('id > 100');
```

2.

```
// Only objects of the specified types will be displayed on the map.
objectManager.setFilter('properties.type == "cafe" || properties.type == "pharmacy");
```

3.

```
// You can define a filter function.
objectManager.setFilter(function (object) {
    return object.properties.name != 'The one who cannot be shown.';
});
```

## objectManager

### objectManager.addon

#### objectManager.addon.clustersBalloon

**Note:** The constructor of the `objectManager.addon.clustersBalloon` class is hidden, as this class is not intended for autonomous initialization.

Static object.

#### Methods

#### Example:

```
&lt;script src="http://api-maps.yandex.ru/2.1/?
lang=ru_RU&load=Map,ObjectManager,objectManager.addon.clustersBalloon" type="text/javascript">&lt;/
script>
```

#### Methods

Name	Returns	Description
<code>get()</code>	<code>IPopupManager</code>	Returns balloon manager of the object manager.

#### Methods details

##### get

```
{IPopupManager} get ()
```

**Returns** balloon manager of the object manager.

#### Example:

```
ymaps.objectManager.addon.clustersBalloon.get(myMap)
```

#### objectManager.addon.clustersHint

**Note:** The constructor of the `objectManager.addon.clustersHint` class is hidden, as this class is not intended for autonomous initialization.

Static object.

#### Methods

**Example:**

```
&lt;script src=&quot;http://api-maps.yandex.ru/2.1/?
lang=ru_RU&load=Map,ObjectManager,objectManager.addon.clustersHint&quot;;type=&quot;text/javascript&quot;;&gt;&lt;/
script&gt;
```

**Methods**

Name	Returns	Description
<a href="#">get()</a>	<a href="#">IPopupManager</a>	Returns hint manager <a href="#">ObjectManager</a> .

**Methods details****get**

```
{IPopupManager} get ()
```

Returns hint manager [ObjectManager](#).

**Example:**

```
ymaps.objectManager.addon.clustersHint.get (myMap)
```

**objectManager.addon.objectsBalloon**

**Note:** The constructor of the `objectManager.addon.objectsBalloon` class is hidden, as this class is not intended for autonomous initialization.

Static object.

**Methods****Example:**

```
&lt;script src=&quot;http://api-maps.yandex.ru/2.1/?
lang=ru_RU&load=Map,ObjectManager,objectManager.addon.objectsBalloon&quot;;type=&quot;text/javascript&quot;;&gt;&lt;/
script&gt;
```

**Methods**

Name	Returns	Description
<a href="#">get(collection)</a>	<a href="#">IPopupManager</a>	Returns balloon manager of the object manager.

**Methods details****get**

```
{IPopupManager} get (collection)
```

Returns balloon manager of the object manager.

**Parameters:**

Parameter	Default value	Description
<a href="#">collection</a> *	—	Type: <a href="#">objectManager.ObjectCollection</a>  Collection

\* Mandatory parameter/option.

#### Example:

```
ymaps.objectManager.addon.objectsBalloon.get(myMap)
```

### objectManager.addon.objectsHint

**Note:** The constructor of the `objectManager.addon.objectsHint` class is hidden, as this class is not intended for autonomous initialization.

Static object.

#### Methods

#### Example:

```
&lt;script src=&quot;http://api-maps.yandex.ru/2.1/?
lang=ru_RU&load=Map,ObjectManager,objectManager.addon.objectsHint&quot;;type=&quot;text/javascript&quot;;&gt;&lt;/
script&gt;
```

#### Methods

Name	Returns	Description
<a href="#">get()</a>	<a href="#">IPopupManager</a>	Returns hint manager <a href="#">ObjectManager</a> .

#### Methods details

##### get

```
{IPopupManager} get ()
```

Returns hint manager [ObjectManager](#).

#### Example:

```
ymaps.objectManager.addon.objectsHint.get(myMap)
```

## objectManager.Balloon

Extends [IBalloonManager](#).

Manager for the collection balloon `ObjectManager`. Allows to manage the object's balloon by opening it and hiding it. It uses the map balloon manager [map.Balloon](#). Collections of objects in [ObjectManager](#) contain an instance of this class, accessible as `myObjectManager.objects.balloon` and `myObjectsLayer.clusters.balloon`. Don't create new instances of this class unless necessary.

See [Balloon](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
objectManager.Balloon(collection)
```

#### Parameters:

Parameter	Default value	Description
<a href="#">collection</a> *	—	Type: <a href="#">IReadOnlyCollection</a> Collection of objects.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

## Events

Name	Description
<a href="#">autopanbegin</a>	Start of automatic shifting of the map center initiated by the autoPan method. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>target - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> Inherited from <a href="#">IBalloonManager</a> .
<a href="#">autopanend</a>	End of automatic shifting of the map center initiated by the autoPan method. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>target - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> Inherited from <a href="#">IBalloonManager</a> .
<a href="#">beforeuserclose</a>	The event which precedes <a href="#">Balloon.event:userclose</a> . Allows you to cancel the user's action by calling the preventDefault method. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>target - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> Inherited from <a href="#">IBalloonManager</a> .
<a href="#">close</a>	Closing the info object. Names of fields available via <a href="#">Event.get</a> : <ul style="list-style-type: none"> <li>target - Reference to the object where the closing occurred.</li> </ul> Inherited from <a href="#">IPopupManager</a> .
<a href="#">open</a>	Opening the info object. Names of fields available via <a href="#">Event.get</a> : <ul style="list-style-type: none"> <li>target - Reference to the object where the opening occurred.</li> </ul> Inherited from <a href="#">IPopupManager</a> .
<a href="#">userclose</a>	Balloon closed by the user. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>target - Reference to the <a href="#">IBalloonOwner</a> object.</li> </ul> Inherited from <a href="#">IBalloonManager</a> .

## Methods

Name	Returns	Description
<a href="#">autoPan()</a>	<a href="#">vow.Promise</a>	Moves the map so that the balloon is visible. Inherited from <a href="#">IBalloonManager</a> .
<a href="#">close([force])</a>	<a href="#">vow.Promise</a>	Closes an open balloon.

Name	Returns	Description
<a href="#">destroy()</a>		Disables the info object manager.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">getData()</a>	Object null	Returns hash describing the object the balloon is open on, or null if the balloon was not opened.
<a href="#">getOptions()</a>	<a href="#">IOptionManager</a>  null	Returns the options manager or 'null'.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object to return the overlay.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	Returns the overlay, if one exists.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">getPosition()</a>	Number[] null	Returns the coordinates of the info object or 'null'.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">isOpen(id)</a>	Boolean	A method that detects whether the balloon is open on the object with the passed identifier.
<a href="#">open(objectId, anchorPixelPosition)</a>	<a href="#">vow.Promise</a>	Opens the balloon on the object with the passed identifier.
<a href="#">setData(objectData)</a>	<a href="#">vow.Promise</a>	Sets new data for displaying the balloon.
<a href="#">setOptions(options)</a>	<a href="#">vow.Promise</a>	Defines new options for the info object.  Inherited from <a href="#">IPopupManager</a> .
<a href="#">setPosition(position)</a>	<a href="#">vow.Promise</a>	Specifies a new position for the info object.  Inherited from <a href="#">IPopupManager</a> .

## Methods details

### close

```
{vow.Promise} close([force])
```

Closes an open balloon.

**Returns** Promise object.

**Parameters:**

Parameter	Default value	Description
<code>force</code>	<code>false</code>	Type: Boolean Instant closure.

**Example:**

```
// Closing all balloons on the layer.
objectManager.objects.balloon.close();
objectManager.clusters.balloon.close();
```

**getData**

```
{Object|null} getData()
```

**Returns** hash describing the object the balloon is open on, or null if the balloon was not opened.

**Example:**

```
var cluster = objectManager.clusters.balloon.getData();
if (cluster) {
  alert('The balloon for a cluster of ' + cluster.properties.geoObjects.length + ' objects is currently
  opened.');
```

**isOpen**

```
{Boolean} isOpen(id)
```

A method that detects whether the balloon is open on the object with the passed identifier.

**Returns** balloon state: open/closed.

**Parameters:**

Parameter	Default value	Description
<code>id</code> *	—	Type: Object Object ID.

\* Mandatory parameter/option.

**Example:**

```
// Closing the ballon after the second click on the placemark.
objectManager.objects.options.set('hideIconOnBalloonOpen', false);
objectManager.objects.events.add('click', function (e) {
  var objectId = e.get('objectId');
  if (objectManager.objects.balloon.isOpen(objectId)) {
    objectManager.objects.balloon.close();
  }
});
```

**open**

```
{vow.Promise} open(objectId, anchorPixelPosition)
```

Opens the balloon on the object with the passed identifier.

**Returns** Promise object.

**Parameters:**

Parameter	Default value	Description
<code>objectId *</code>	—	Type: Object  ID of the object to open the balloon on.
<code>anchorPixelPosition *</code>	—	Type:

\* Mandatory parameter/option.

**Example:**

```
// Loading the object data before opening the balloon.
// Disabling automatic balloon opening by clicking on the object.
objectManager.options.set('geoObjectOpenBalloonOnClick', false);
objectManager.objects.events.add('click', function (e) {
  var objectId = e.get('objectId');
  // Preloading the object data after the object was clicked.
  loadObjectData(objectId).then(function (data) {
    // As soon as the data is ready, we're assigning it to the object and opening the balloon.
    objectManager.objects.getById(objectId).properties = data;
    objectManager.objects.balloon.open(objectId);
  });
});
```

**setData**

```
{vow.Promise} setData(objectData)
```

Sets new data for displaying the balloon.

Returns Promise object.

**Parameters:**

Parameter	Default value	Description
<code>objectData *</code>	—	Type: Object  Hash with a description of the object to open the balloon on. Corresponds to the object description that is input to <code>ObjectManager.add</code> .

\* Mandatory parameter/option.

**Example:**

```
// Preloading the object data after the ballon is opened on the object.
objectManager.objects.events.add('balloonopen', function (e) {
  var objectId = e.get('objectId'),
      object = objectManager.objects.getById(objectId);
  if (!object.properties) {
    loadObjectData(objectId).then(function (data) {
      if (objectManager.balloon.isOpen(objectid) {
        object.properties = data;
        objectManager.objects.balloon.setData(object);
      }
    });
  }
});
```

**objectManager.ClusterCollection**

Extends [ICustomizable](#), [IEventEmitter](#).

Collection of clusters generated by [ObjectManager](#). Clusters are added to and deleted from the collection automatically, and are read-only. The cluster object is a JSON structure the same as the objects in the layer. Cluster object field:

- `id` - Unique cluster ID.
- `geometry` - Description of the cluster geometry.
- `properties` - Description of the cluster data. The `properties.geoObjects` field stores an array of objects that are included in the cluster.
- `options` - Cluster options. Optional field.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

## Constructor

```
objectManager.ClusterCollection()
```

## Fields

Name	Type	Description
<a href="#">balloon</a>	<a href="#">objectManager.Balloon</a>	Cluster balloon in the manager.
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">hint</a>	<a href="#">objectManager.Hint</a>	Object hint in the <a href="#">ObjectManager</a> . Names of fields available via <a href="#">Event.get</a> : <ul style="list-style-type: none"> <li>• <code>objectId</code> - ID of the object where the hint was shown.</li> </ul>
<a href="#">options</a>	<a href="#">option.Manager</a>	Options manager. Names of fields that are available via the <code>option.Manager#get</code> method: <ul style="list-style-type: none"> <li>• <code>hasBalloon</code> - Indicates whether the collection has the <code>.balloon</code> field. If a balloon doesn't need to be opened when clicking the cluster, we recommend setting this option to the "false" value to avoid unnecessary initializations.</li> <li>• <code>hasHint</code> - Indicates whether the collection has the <code>.hint</code> field. If a popup hint doesn't need to be displayed when the cluster is pointed at, we recommend setting this option to the "false" value to avoid unnecessary initializations.</li> <li>• <code>hideIconOnBalloonOpen</code> - Hide the icon when opening the balloon. Default value: true.</li> <li>• <code>openBalloonOnClick</code> - Option that allows you to forbid opening the balloon when clicking on a cluster. By default, opening the balloon is allowed.</li> <li>• <code>openHintOnHover</code> - Option that allows you to forbid displaying the popup hint when the cluster is pointed at. By default, showing hints is allowed.</li> </ul>
<a href="#">overlays</a>	<a href="#">objectManager.OverlayCollection</a>	Collection of cluster overlays. All events, with the exception of "add" and "remove" events, propagate from the collection of overlays to the collection of clusters.
<a href="#">state</a>	<a href="#">data.Manager</a>	State of the collection of clusters. Defined by the following fields: <ul style="list-style-type: none"> <li>• <code>activeObject</code> - JSON description of the object selected in the cluster balloon.</li> </ul>

**Events**

Name	Description
<a href="#">add</a>	<p>Adds a cluster to the collection. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>objectId</code> - ID of the added object.</li> <li>• <code>child</code> - The added object.</li> </ul>
<a href="#">clusteroptionschange</a>	<p>Modification of cluster options via the <a href="#">objectManager.ClusterCollection.setClusterOptions</a> method. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>objectId</code> - ID of the cluster that had options modified.</li> </ul>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">remove</a>	<p>Deletes a cluster from the collection. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>objectId</code> - ID of the deleted object.</li> <li>• <code>child</code> - The deleted object.</li> </ul>

**Methods**

Name	Returns	Description
<a href="#">each(callback, context)</a>		
<a href="#">getAll()</a>	<code>Object[]</code>	Returns array of objects contained in the collection.
<a href="#">getById(id)</a>	<code>Object null</code>	Returns cluster object with the specified ID, or null if this cluster does not exist.
<a href="#">getIterator()</a>	<code>Iterator</code>	Returns iterator for the collection.
<a href="#">getLength()</a>	<code>Number</code>	Returns the number of objects in the collection.
<a href="#">getObjectManager()</a>	<code>ObjectManager</code>	Returns the parent layer of objects in the collection.
<a href="#">setClusterOptions(objectId, options)</a>	<code>objectManager.ObjectCollection</code>	Returns self-reference.

**Fields details****balloon**

```
{objectManager.Balloon} balloon
```

Cluster balloon in the manager.

**hint**

```
{objectManager.Hint} hint
```

Object hint in the [ObjectManager](#). Names of fields available via [Event.get](#):

- `objectId` - ID of the object where the hint was shown.

## options

```
{option.Manager} options
```

Options manager. Names of fields that are available via the `option.Manager#get` method:

- `hasBalloon` - Indicates whether the collection has the `.balloon` field. If a balloon doesn't need to be opened when clicking the cluster, we recommend setting this option to the "false" value to avoid unnecessary initializations.
- `hasHint` - Indicates whether the collection has the `.hint` field. If a popup hint doesn't need to be displayed when the cluster is pointed at, we recommend setting this option to the "false" value to avoid unnecessary initializations.
- `hideIconOnBalloonOpen` - Hide the icon when opening the balloon. Default value: true.
- `openBalloonOnClick` - Option that allows you to forbid opening the balloon when clicking on a cluster. By default, opening the balloon is allowed.
- `openHintOnHover` - Option that allows you to forbid displaying the popup hint when the cluster is pointed at. By default, showing hints is allowed.

### Example:

```
objectManager.objects.options.set({
  preset: 'islands#greenDotIcon',
  hintContentLayout: ymaps.templateLayoutFactory.createClass('{{properties.name}}')
});
```

## overlays

```
{objectManager.OverlayCollection} overlays
```

Collection of cluster overlays. All events, with the exception of "add" and "remove" events, propagate from the collection of overlays to the collection of clusters.

### Example:

```
// Changing the color of the cluster icon when moused over.
objectManager.clusters.events.add(['mouseenter', 'mouseleave'], function (e) {
  var objectId = e.get('objectId');
  var overlay = objectManager.clusters.overlays.getById(objectId);
  if (e.get('type') == 'mouseenter') {
    setRedColor(objectId);
    overlay.events.add('mapchange', onMapChange);
  } else {
    setGreenColor(objectId);
    overlay.events.remove('mapchange', onMapChange);
  }
});

function onMapChange (e) {
  setGreenColor(objectManager.clusters.overlays.getId(e.get('target')));
}

function setGreenColor (objectId) {
  objectManager.clusters.setClusterOptions(objectId, {
    preset: 'islands#greenClusterIcons'
  });
}

function setRedColor (objectId) {
  objectManager.clusters.setClusterOptions(objectId, {
    preset: 'islands#redClusterIcons'
  });
}
```

## state

```
{data.Manager} state
```

State of the collection of clusters. Defined by the following fields:

- `activeObject` - JSON description of the object selected in the cluster balloon.

**Example:**

```
// Opening the cluster balloon with the selected object.
var objectState = objectManager.getObjectState(myObjects[i]);
if (objectState.isClustered) {
  objectManager.clusters.state.set('activeObject', myObjects[i]);
  objectManager.clusters.balloon.open(objectState.cluster.id);
}
```

**Events details****add**

Adds a cluster to the collection. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `objectId` - ID of the added object.
- `child` - The added object.

**clusteroptionschange**

Modification of cluster options via the [objectManager.ClusterCollection.setClusterOptions](#) method. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `objectId` - ID of the cluster that had options modified.

**remove**

Deletes a cluster from the collection. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `objectId` - ID of the deleted object.
- `child` - The deleted object.

**Methods details****each**

```
{ } each(callback, context)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">callback</a> *	—	Type: Function  Callback function that the collection objects are passed to.
<a href="#">context</a> *	—	Type: Object  Context for the callback.

\* Mandatory parameter/option.

**Example:**

```
var clusterizedObjectsCounter = 0;
objectManager.clusters.each(function (cluster) {
  clusterizedObjectsCounter += cluster.properties.geoObjects.length;
});
alert('The map shows ' + clusterizedObjectsCounter + ' clusterized objects.');
```

## getAll

```
{Object[]} getAll()
```

Returns array of objects contained in the collection.

### Example:

```
var clusterArray = objectManager.clusters.getAll();
```

## getById

```
{Object|null} getById(id)
```

Returns cluster object with the specified ID, or null if this cluster does not exist.

### Parameters:

Parameter	Default value	Description
<code>id</code> *	—	Type: String Cluster ID.

\* Mandatory parameter/option.

### Example:

```
// Making the cluster color change if it has more than 20 objects.
objectManager.clusters.events.add('add', function (e) {
  var cluster = objectManager.clusters.getById(e.get('objectId'));
  var objects = cluster.properties.geoObjects;
  if (objects.length > 20) {
    objectManager.clusters.setClusterOptions(cluster.id, {
      preset: 'islands#redClusterIcons'
    });
  }
});
```

## getIterator

```
{Iterator} getIterator()
```

Returns iterator for the collection.

### Example:

```
var clusterizedObjectsCounter = 0;
var it = objectManager.clusters.getIterator();
var cluster;
while ((cluster = it.getNext()) != it.STOP_ITERATION)
  clusterizedObjectsCounter += cluster.pProperties.geoObjects.length;
});
alert('The map displays ' + clusterizedObjectsCounter + ' clusterized objects.');
```

## getLength

```
{Number} getLength()
```

Returns the number of objects in the collection.

### Example:

```
alert('The map displays ' + objectManager.clusters.getLength() + ' clusters.');
```

## getManager

```
{ObjectManager} getManager()
```

Returns the parent layer of objects in the collection.

## setClusterOptions

```
{objectManager.ObjectCollection} setClusterOptions(objectId, options)
```

Returns self-reference.

### Parameters:

Parameter	Default value	Description
<code>objectId</code> *	—	Type: String Cluster ID.
<code>options</code> *	—	Type: Object Object with cluster options.

\* Mandatory parameter/option.

### Example:

```
// Making the cluster color change if it has more than 20 objects.
objectManager.clusters.events.add('add', function (e) {
  var cluster = objectManager.clusters.getById(e.get('objectId'));
  var objects = cluster.properties.geoObjects;
  if (objects.length > 20) {
    objectManager.clusters.setClusterOptions(cluster.id, {
      preset: 'islands#redClusterIcons'
    });
  }
});
```

## objectManager.Hint

Extends [IHintManager](#).

Manager of the hint on an object layer. Allows to manage the hint on an object layer by opening it and hiding it. It uses the map hint manager [map.Hint](#) inside itself. Object layers contains instances of this class, accessible as `myObjectManager.objects.hint` and `myObjectManager.clusters.hint`. Don't create new instances of this class unless necessary.

See [Hint](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
objectManager.Hint(collection)
```

### Parameters:

Parameter	Default value	Description
<code>collection</code> *	—	Type: IReadOnlyCollection Object layer.

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

**Events**

Name	Description
<a href="#">close</a>	Closing the info object. Names of fields available via <a href="#">Event.get</a> : <ul style="list-style-type: none"> <li><code>target</code> - Reference to the object where the closing occurred.</li> </ul> Inherited from <a href="#">IPopupManager</a> .
<a href="#">open</a>	Opening the info object. Names of fields available via <a href="#">Event.get</a> : <ul style="list-style-type: none"> <li><code>target</code> - Reference to the object where the opening occurred.</li> </ul> Inherited from <a href="#">IPopupManager</a> .

**Methods**

Name	Returns	Description
<a href="#">close</a> ( <a href="#">[force]</a> )	<a href="#">vow.Promise</a>	Hides the popup hint.
<a href="#">destroy</a> ()		Disables the info object manager. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getData</a> ()	Object null	Returns hash describing the object the hint is shown on, or null if the hint was not shown.
<a href="#">getOptions</a> ()	<a href="#">IOptionManager</a>  null	Returns the options manager or 'null'. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOverlay</a> ()	<a href="#">vow.Promise</a>	Returns the promise object to return the overlay. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getOverlaySync</a> ()	<a href="#">IOverlay</a>  null	Returns the overlay, if one exists. Inherited from <a href="#">IPopupManager</a> .
<a href="#">getPosition</a> ()	Number[] null	Returns the coordinates of the info object or 'null'. Inherited from <a href="#">IPopupManager</a> .
<a href="#">isOpen</a> ( <a href="#">id</a> )	Boolean	A method that determines whether the popup hint was shown on the object with the passed ID.

Name	Returns	Description
<code>open(objectId[, position])</code>	<code>vow.Promise</code>	Displays the popup hint on the object with the passed ID.
<code>setData(objectData)</code>	<code>vow.Promise</code>	Sets new data for displaying the popup hint.
<code>setOptions(options)</code>	<code>vow.Promise</code>	Defines new options for the info object.  Inherited from <a href="#">IPopupManager</a> .
<code>setPosition(position)</code>	<code>vow.Promise</code>	Specifies a new position for the info object.  Inherited from <a href="#">IPopupManager</a> .

## Methods details

### close

```
{vow.Promise} close([force])
```

Hides the popup hint.

**Returns** Promise object.

#### Parameters:

Parameter	Default value	Description
<code>force</code>	<code>false</code>	Type: Boolean  Instant closure.

#### Example:

```
// Closing all hints on a layer.
objectManager.objects.hint.close();
objectManager.clusters.hint.close();
```

### getData

```
{Object|null} getData()
```

**Returns** hash describing the object the hint is shown on, or null if the hint was not shown.

#### Example:

```
var cluster = objectManager.clusters.hint.getData();
if (cluster) {
  alert('Cluster displays a hint.');
```

### isOpen

```
{Boolean} isOpen(id)
```

A method that determines whether the popup hint was shown on the object with the passed ID.

**Returns** hint state: shown/hidden.

#### Parameters:

Parameter	Default value	Description
<code>id</code> *	—	Type: Object Object ID.

\* Mandatory parameter/option.

#### Example:

```
// Closing the hint when the object is clicked.
objectManager.objects.add('click', function (e) {
  var objectId = e.get('objectId');
  if (objectManager.objects.hint.isOpen(objectId)) {
    objectManager.objects.hint.close();
  }
});
```

## open

```
{vow.Promise} open(objectId[, position])
```

Displays the popup hint on the object with the passed ID.

**Returns** Promise object.

#### Parameters:

Parameter	Default value	Description
<code>objectId</code> *	—	Type: Object ID of the object to open the hint on.
<code>position</code>	—	Type: Number[] The position for showing the popup hint, in global pixel coordinates. If the value is not specified, the hint will appear at the geometric center of the object.

\* Mandatory parameter/option.

#### Example:

```
objectManager.clusters.hint.open(objectId);
```

## setData

```
{vow.Promise} setData(objectData)
```

Sets new data for displaying the popup hint.

**Returns** Promise object.

#### Parameters:

Parameter	Default value	Description
<a href="#">objectData</a> *	—	Type: Object  Hash with a description of the object to open the hint on. Corresponds to the object description that is input to <code>ObjectManager.add</code> .

\* Mandatory parameter/option.

#### Example:

```
objectManager.objects.hint.setData(objectManager.objects.getById(objectId));
```

## objectManager.ObjectCollection

Extends [ICollection](#), [ICustomizable](#).

Collection of objects added to the layer.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
objectManager.ObjectCollection()
```

### Fields

Name	Type	Description
<a href="#">balloon</a>	<a href="#">objectManager.Balloon</a>	Object balloon in the manager.
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .
<a href="#">hint</a>	<a href="#">objectManager.Hint</a>	Object hint in the <a href="#">ObjectManager</a> .
<a href="#">options</a>	<a href="#">option.Manager</a>	Options manager. Names of fields that are available via the <code>option.Manager#get</code> method: <ul style="list-style-type: none"> <li><code>hasBalloon</code> - Indicates whether the collection has the <code>.balloon</code> field. If a balloon doesn't need to be opened when clicking the object, we recommend setting this option to the "false" value to avoid unnecessary initializations.</li> <li><code>hasHint</code> - Indicates whether the collection has the <code>.hint</code> field. If a popup hint doesn't need to be displayed when the object is pointed at, we recommend setting this option to the "false" value to avoid unnecessary initializations.</li> <li><code>hideIconOnBalloonOpen</code> - Hide the icon when opening the balloon. Default value: true.</li> <li><code>openBalloonOnClick</code> - Show the balloon when clicking the object. Default value: true.</li> </ul>
<a href="#">overlays</a>	<a href="#">objectManager.OverlayCollection</a>	Collection of overlays of singular objects. All events, with the exception of "add" and "remove" events, propagate from the collection of overlays to the collection of objects.

## Events

Name	Description
<a href="#">add</a>	<p>Adds an object to the collection. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>objectId</code> - ID of the added object.</li> <li>• <code>child</code> - The added object.</li> </ul>
<a href="#">objectoptionschange</a>	<p>Modification of object options via the <a href="#">objectManager.ObjectCollection.setObjectOptions</a> method. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>objectId</code> - ID of the object that had options modified.</li> </ul>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">remove</a>	<p>Removes an object from the collection. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>objectId</code> - ID of the deleted object.</li> <li>• <code>child</code> - The deleted object.</li> </ul>

## Methods

Name	Returns	Description
<a href="#">add(data)</a>	<a href="#">objectManager.ObjectCollection</a>	This method completely duplicates the logic of <a href="#">ObjectManager.add</a> .
<a href="#">each(callback, context)</a>		A method that calls the passed handler function for all the items in the collection.
<a href="#">getAll()</a>	<code>Object[]</code>	Returns array of objects contained in the collection.
<a href="#">getById(id)</a>	<code>Object null</code>	Returns the object, or null if an object with the passed ID does not exist.
<a href="#">getIterator()</a>	<a href="#">Iterator</a>	Returns iterator for the collection.
		Inherited from <a href="#">ICollection</a> .
<a href="#">getLength()</a>	<code>Number</code>	Returns the number of objects in the collection.
<a href="#">getObjectManager()</a>	<a href="#">ObjectManager</a>	Returns the parent layer of objects in the collection.
<a href="#">remove(data)</a>	<a href="#">objectManager.ObjectCollection</a>	This method completely duplicates the logic of <a href="#">ObjectManager.remove</a> .
<a href="#">removeAll()</a>	<a href="#">objectManager.ObjectCollection</a>	Deletes all the items from a collection.

Name	Returns	Description
<code>setObjectOptions(objectId, options)</code>	<code>objectManager.ObjectCollection</code>	A method that allows to dynamically update object options. This method should be used when you want new options to be immediately applied to the object representation on the map. Note that the manager ignores the 'visible' option.

## Fields details

### balloon

```
{objectManager.Balloon} balloon
```

Object balloon in the manager.

### hint

```
{objectManager.Hint} hint
```

Object hint in the [ObjectManager](#).

### options

```
{option.Manager} options
```

Options manager. Names of fields that are available via the `option.Manager#get` method:

- `hasBalloon` - Indicates whether the collection has the `.balloon` field. If a balloon doesn't need to be opened when clicking the object, we recommend setting this option to the "false" value to avoid unnecessary initializations.
- `hasHint` - Indicates whether the collection has the `.hint` field. If a popup hint doesn't need to be displayed when the object is pointed at, we recommend setting this option to the "false" value to avoid unnecessary initializations.
- `hideIconOnBalloonOpen` - Hide the icon when opening the balloon. Default value: true.
- `openBalloonOnClick` - Show the balloon when clicking the object. Default value: true.

### Example:

```
// Creating styles for individual objects within the collection.
objectManager.objects.options.set({
  preset: 'islands#redIcon',
  hasBalloon: false,
  zIndex: 500
});
```

### overlays

```
{objectManager.OverlayCollection} overlays
```

Collection of overlays of singular objects. All events, with the exception of "add" and "remove" events, propagate from the collection of overlays to the collection of objects.

### Example:

```
objectManager.objects.overlays.events.add('add', function (e) {
  alert('The object ' + e.get('objectId') + ' is shown on the map.');
```

## Events details

### add

Adds an object to the collection. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `objectId` - ID of the added object.
- `child` - The added object.

### objectoptionschange

Modification of object options via the [objectManager.ObjectCollection.setObjectOptions](#) method. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `objectId` - ID of the object that had options modified.

### remove

Removes an object from the collection. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `objectId` - ID of the deleted object.
- `child` - The deleted object.

## Methods details

### add

```
{objectManager.ObjectCollection} add(data)
```

This method completely duplicates the logic of [ObjectManager.add](#).

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>data</code> *	—	Type: <code>Object Object[] String</code> Objects to add to the layer.

\* Mandatory parameter/option.

**Example:**

```
objectManager.objects.add({
  type: 'Feature',
  geometry: {
    type: 'Point',
    coordinates: [55.33, 36.64]
  }
});
```

### each

```
{ } each(callback, context)
```

A method that calls the passed handler function for all the items in the collection.

**Parameters:**

Parameter	Default value	Description
<code>callback *</code>	—	Type: Function  Handler function that the collection objects are passed to.
<code>context *</code>	—	Type: Object  Context for the handler function.

\* Mandatory parameter/option.

#### Example:

```
var singleCounter = 0;
var clusterCounter = 0;
objectManager.objects.each(function (object) {
  var objectState = objectManager.getObjectState(object.id);
  if (objectState.isClustered) {
    clusterCounter++;
  } else {
    if (objectState.isShown) {
      singleCounter++;
    }
  }
});
alert('Number of single placemarks shown: ' + singleCounter);
alert('Number of placemarks shown in clusters: ' + clusterCounter);
```

#### getAll

```
{Object[]} getAll()
```

Returns array of objects contained in the collection.

#### getById

```
{Object|null} getById(id)
```

Returns the object, or null if an object with the passed ID does not exist.

#### Parameters:

Parameter	Default value	Description
<code>id *</code>	—	Type: Number  Object ID.

\* Mandatory parameter/option.

#### Example:

```
objectManager.objects.add('click', function (e) {
  var objectId = e.get('objectId');
  var object = objectManager.objects.getById(objectId);
});
```

#### getLength

```
{Number} getLength()
```

Returns the number of objects in the collection.

**Example:**

```
alert('Number of objects in the layer: ' + objectManager.objects.getLength());
```

**getObjectManager**

```
{objectManager} getObjectManager ()
```

**Returns** the parent layer of objects in the collection.

**remove**

```
{objectManager.ObjectCollection} remove (data)
```

This method completely duplicates the logic of [objectManager.remove](#).

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>data</code> *	—	Type: Object Object[] String The objects to delete.

\* Mandatory parameter/option.

**Example:**

```
// Removing the objects with the IDs 34 and 25.
objectManager.objects.remove([34, 25]);
```

**removeAll**

```
{objectManager.ObjectCollection} removeAll ()
```

Deletes all the items from a collection.

**Returns** self-reference.

**Example:**

```
objectManager.objects.removeAll ();
```

**setObjectOptions**

```
{objectManager.ObjectCollection} setObjectOptions (objectId, options)
```

A method that allows to dynamically update object options. This method should be used when you want new options to be immediately applied to the object representation on the map. Note that the manager ignores the 'visible' option.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>objectId</code> *	—	Type: Object ID of the object to set options for.

Parameter	Default value	Description
<code>options</code> *	—	Type: Object New object options.

\* Mandatory parameter/option.

#### Example:

```
// Changing the icon color when moused over.
objectManager.objects.events.add('mouseenter', function (e) {
  var objectId = e.get('objectId');
  var overlay = objectManager.objects.overlays.getById(objectId);
  setRedColor(objectId);
  overlay.events.add('mapchange', setGreenColor);
});

objectManager.objects.events.add('mouseleave', function (e) {
  var objectId = e.get('objectId');
  var overlay = objectManager.objects.overlays.getById(objectId);
  setGreenColor(objectId);
  overlay.events.remove('mapchange', setGreenColor);
});

function setGreenColor (objectId) {
  objectManager.objects.setObjectOptions(objectId, {
    preset: 'islands#greenIcon'
  });
}

function setRedColor (objectId) {
  objectManager.objects.setClusterOptions(objectId, {
    preset: 'islands#redIcon'
  });
}
```

## objectManager.OverlayCollection

Extends [ICustomizable](#), [IEventEmitter](#).

Collection of overlays.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
objectManager.OverlayCollection()
```

### Fields

Name	Type	Description
<code>events</code>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<code>options</code>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

### Events

Name	Description
<code>add</code>	Adds an overlay to the collection. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>objectId</code> - The ID of the object the overlay belongs to.</li> <li><code>overlay</code> - The added overlay.</li> </ul>

Name	Description
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>objectId</code> - The ID of the object the overlay belongs to.</li><li>• <code>overlay</code> - <code>IOverlay</code> - The overlay at which the event occurred.</li><li>• <code>coords</code> - Geographical coordinates of the point at which the event occurred.</li><li>• <code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li><li>• <code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li><li>• <code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>objectId</code> - The ID of the object the overlay belongs to.</li><li>• <code>overlay</code> - <code>IOverlay</code> - The overlay at which the event occurred.</li><li>• <code>coords</code> - Geographical coordinates of the point at which the event occurred.</li><li>• <code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li><li>• <code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li><li>• <code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>objectId</code> - The ID of the object the overlay belongs to.</li><li>• <code>overlay</code> - <code>IOverlay</code> - The overlay at which the event occurred.</li><li>• <code>coords</code> - Geographical coordinates of the point at which the event occurred.</li><li>• <code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li><li>• <code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li><li>• <code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>

Name	Description
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>objectId</code> - The ID of the object the overlay belongs to.</li><li>• <code>overlay</code> - <code>IOverlay</code> - The overlay at which the event occurred.</li><li>• <code>coords</code> - Geographical coordinates of the point at which the event occurred.</li><li>• <code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li><li>• <code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li><li>• <code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>objectId</code> - The ID of the object the overlay belongs to.</li><li>• <code>overlay</code> - <code>IOverlay</code> - The overlay at which the event occurred.</li><li>• <code>coords</code> - Geographical coordinates of the point at which the event occurred.</li><li>• <code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li><li>• <code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li><li>• <code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>objectId</code> - The ID of the object the overlay belongs to.</li><li>• <code>overlay</code> - <code>IOverlay</code> - The overlay at which the event occurred.</li><li>• <code>coords</code> - Geographical coordinates of the point at which the event occurred.</li><li>• <code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li><li>• <code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li><li>• <code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>

Name	Description
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>objectId</code> - The ID of the object the overlay belongs to.</li><li>• <code>overlay</code> - <code>IOverlay</code> - The overlay at which the event occurred.</li><li>• <code>coords</code> - Geographical coordinates of the point at which the event occurred.</li><li>• <code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li><li>• <code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li><li>• <code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>. Instance of the Event class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>objectId</code> - The ID of the object the overlay belongs to.</li><li>• <code>overlay</code> - <code>IOverlay</code> - The overlay at which the event occurred.</li><li>• <code>coords</code> - Geographical coordinates of the point at which the event occurred.</li><li>• <code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li><li>• <code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li><li>• <code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Names of fields that are available via the <code>IMultiTouchEvent#get</code> method:</p> <ul style="list-style-type: none"><li>• <code>objectId</code> - The ID of the object the overlay belongs to.</li><li>• <code>overlay</code> - <code>IOverlay</code> - The overlay at which the event occurred.</li><li>• <code>coords</code> - Geographical coordinates of the point at which the event occurred.</li><li>• <code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li><li>• <code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li><li>• <code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul>

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Names of fields that are available via the <code>IMultiTouchEvent#get</code> method:</p> <ul style="list-style-type: none"> <li><code>objectId</code> - The ID of the object the overlay belongs to.</li> <li><code>overlay</code> - <code>IOverlay</code> - The overlay at which the event occurred.</li> <li><code>coords</code> - Geographical coordinates of the point at which the event occurred.</li> <li><code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li> <li><code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li> <li><code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li> <li><code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Names of fields that are available via the <code>IMultiTouchEvent#get</code> method:</p> <ul style="list-style-type: none"> <li><code>objectId</code> - The ID of the object the overlay belongs to.</li> <li><code>overlay</code> - <code>IOverlay</code> - The overlay at which the event occurred.</li> <li><code>coords</code> - Geographical coordinates of the point at which the event occurred.</li> <li><code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li> <li><code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li> <li><code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li> <li><code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">remove</a>	<p>Removes an overlay from the collection. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>objectId</code> - The ID of the object the overlay belongs to.</li> <li><code>overlay</code> - The deleted overlay.</li> </ul>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li><code>objectId</code> - The ID of the object the overlay belongs to.</li> <li><code>overlay</code> - <code>IOverlay</code> - The overlay at which the event occurred.</li> <li><code>coords</code> - Geographical coordinates of the point at which the event occurred.</li> <li><code>globalPixels</code> - Coordinates of the event in global pixels from the upper-left corner of the world.</li> <li><code>pagePixels</code> - Coordinates of the event in pixels from the upper-left corner of the page.</li> <li><code>clientPixels</code> - Coordinates of the event in pixels from the upper-left corner of the browser window.</li> <li><code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul>

**Methods**

Name	Returns	Description
<code>each(callback, context)</code>		
<code>getAll()</code>	Object[]	Returns array of objects contained in the collection.
<code>getById(id)</code>	Object null	Returns overlay, or null if an overlay with the passed ID does not exist.
<code>getId(overlay)</code>	Number null	Returns the object ID, or null if the overlay is not contained in the collection.
<code>getIterator()</code>	Iterator	Returns iterator for the collection.
<code>getLength()</code>	Number	Returns the number of objects in the collection.

**Events details****add**

Adds an overlay to the collection. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `objectId` - The ID of the object the overlay belongs to.
- `overlay` - The added overlay.

**click**

Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#). Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `objectId` - The ID of the object the overlay belongs to.
- `overlay` - [IOverlay](#) - The overlay at which the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

**contextmenu**

Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#). Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `objectId` - The ID of the object the overlay belongs to.
- `overlay` - [IOverlay](#) - The overlay at which the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

### **dblclick**

Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#). Instance of the Event class. Names of fields that are available via the [Event.get](#) method:

- `objectId` - The ID of the object the overlay belongs to.
- `overlay` - `IOverlay` - The overlay at which the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

### **mousedown**

Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#). Instance of the Event class. Names of fields that are available via the [Event.get](#) method:

- `objectId` - The ID of the object the overlay belongs to.
- `overlay` - `IOverlay` - The overlay at which the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

### **mouseenter**

Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#). Instance of the Event class. Names of fields that are available via the [Event.get](#) method:

- `objectId` - The ID of the object the overlay belongs to.
- `overlay` - `IOverlay` - The overlay at which the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

### **mouseleave**

Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#). Instance of the Event class. Names of fields that are available via the [Event.get](#) method:

- `objectId` - The ID of the object the overlay belongs to.
- `overlay` - `IOverlay` - The overlay at which the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

**mousemove**

Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. Instance of the Event class. Names of fields that are available via the [Event.get](#) method:

- `objectId` - The ID of the object the overlay belongs to.
- `overlay` - `IOverlay` - The overlay at which the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

**mouseup**

Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#). Instance of the Event class. Names of fields that are available via the [Event.get](#) method:

- `objectId` - The ID of the object the overlay belongs to.
- `overlay` - `IOverlay` - The overlay at which the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

**multitouchend**

End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the `IMultiTouchEvent` interface with information about touches. Names of fields that are available via the `IMultiTouchEvent#get` method:

- `objectId` - The ID of the object the overlay belongs to.
- `overlay` - `IOverlay` - The overlay at which the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

**multitouchmove**

Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the `IMultiTouchEvent` interface with information about touches. Names of fields that are available via the `IMultiTouchEvent#get` method:

- `objectId` - The ID of the object the overlay belongs to.
- `overlay` - `IOverlay` - The overlay at which the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

**multitouchstart**

Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the `IMultiTouchEvent` interface with information about touches. Names of fields that are available via the `IMultiTouchEvent#get` method:

- `objectId` - The ID of the object the overlay belongs to.
- `overlay` - `IOverlay` - The overlay at which the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

**remove**

Removes an overlay from the collection. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- `objectId` - The ID of the object the overlay belongs to.
- `overlay` - The deleted overlay.

**wheel**

Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in [domEvent.manager](#). Instance of the `Event` class. Names of fields that are available via the [Event.get](#) method:

- `objectId` - The ID of the object the overlay belongs to.
- `overlay` - `IOverlay` - The overlay at which the event occurred.
- `coords` - Geographical coordinates of the point at which the event occurred.
- `globalPixels` - Coordinates of the event in global pixels from the upper-left corner of the world.
- `pagePixels` - Coordinates of the event in pixels from the upper-left corner of the page.
- `clientPixels` - Coordinates of the event in pixels from the upper-left corner of the browser window.
- `domEvent` - Source DOM event (as a [DomEvent](#) object), if there is one.

**Methods details****each**

```
{ } each(callback, context)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">callback</a> *	—	Type: Function  Callback function that the collection objects are passed to.
<a href="#">context</a> *	—	Type: Object  Context for the callback.

\* Mandatory parameter/option.

**Example:**

```
objectManager.clusters.overlays.each(function (overlay) {
  overlay.options.set('cursor', 'help');
});
```

```
});
```

## getAll

```
{Object[]} getAll()
```

Returns array of objects contained in the collection.

### Example:

```
var clusterOverlayArray = objectManager.clusters.overlays.getAll();
```

## getById

```
{Object|null} getById(id)
```

Returns overlay, or null if an overlay with the passed ID does not exist.

### Parameters:

Parameter	Default value	Description
<code>id</code> *	—	Type: Number  ID of the object the overlay belongs to.

\* Mandatory parameter/option.

### Example:

```
objectManager.objects.add('mouseenter', function (e) {
  var objectId = e.get('objectId');
  var overlay = objectManager.objects.overlays.getById(objectId);
  overlay.options.set('zIndex', 100);
});
```

## getId

```
{Number|null} getId(overlay)
```

Returns the object ID, or null if the overlay is not contained in the collection.

### Parameters:

Parameter	Default value	Description
<code>overlay</code> *	—	Type: <a href="#">IOverlay</a>  Overlay.

\* Mandatory parameter/option.

### Example:

```
objectManager.objects.overlays.each(function (overlay) {
  var objectId = objectManager.objects.overlays.getId(overlay);
  objectManager.objects.setObjectOptions(objectId, {
    preset: 'islands#redIcon'
  });
});
```

## getIterator

```
{Iterator} getIterator()
```

**Returns** iterator for the collection.

**Example:**

```
var it = objectManager.objects.overlays.getIterator();
var overlay;
while ((overlay = it.getNext()) != it.STOP_ITERATION) {
    overlay.options.set('zIndex', 10);
}
```

### getLength

```
{Number} getLength()
```

**Returns** the number of objects in the collection.

**Example:**

```
var objectsNumber = objectManager.objects.getLength(),
    overlaysNumber = objectManager.objects.overlays.getNumber();
alert('Now the map shows ' + overlaysNumber + ' of ' + objectsNumber.);
```

## option

### option.Manager

Extends [IOptionManager](#).

Options manager. For setting and getting option values by a string key, as well as allowing option values in the context of the existing hierarchy of options managers.

The special "preset" key is for making a set of options by default for this manager. The value of the "preset" option can be a hash with the format {"option name": "option value"}, or a string ID for a hash of options in the [option.presetStorage](#) storage. This hash of options can also contain a field named "preset", which allows for inheritance of option values from other sets of options.

When searching for values in the hierarchy, first the options themselves are checked, then the options set using the "preset" key, and after that the parent is accessed, if there is one.

To track changes to certain options, you can use [Monitor](#).

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
option.Manager([options[, parent[, name]])
```

Creates an options manager.

**Parameters:**

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object Hash of options.
<a href="#">parent</a>	—	Type: <a href="#">IOptionManager</a> Parent options manager.

Parameter	Default value	Description
<a href="#">name</a>	—	Type: String  Name of the options manager.

**Examples:****1.**

```
// Example of building a hierarchy of options managers.
var parentManager = new ymaps.option.Manager({
  key1: '123'
});
var childManager = new ymaps.option.Manager({
  key2: '234'
}, parentManager);
// Outputs 123. The value is taken from manager1.
alert(childManager.get('key1'));
// Outputs 234. The value is taken from manager2.
alert(childManager.get('key2'));
// Overriding the option.
childManager.set('key1', '345');
// Outputs 345. The value is taken from manager2.
alert(childManager.get('key1'));
// Outputs 123. The value is taken from manager1.
alert(parentManager.get('key1'));
```

**2.**

```
// Example using the "preset" option.
var optionManager = new ymaps.option.Manager({
  preset: 'islands#blueIcon'
});
var subOptionManager = new ymaps.option.Manager();
// There is no data, because subOptionManager is empty.
alert(subOptionManager.get('iconImageSize'));
// Binding two managers.
subOptionManager.setParent(optionManager);
// [37, 42] - value is taken from the preset in the parent manager.
alert(subOptionManager.get('iconImageSize'));
// Overriding the value of iconImageSize on the level of subOptionManager.
subOptionManager.set('iconImageSize', [10, 12]);
// [10, 12] - value is taken from subOptionManager.
alert(subOptionManager.get('iconImageSize'));
// Canceling the override of iconImageSize.
subOptionManager.unset('iconImageSize');
// [37, 42] - value is again taken from the preset in the parent manager.
alert(subOptionManager.get('iconImageSize'));
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager for the object.  Inherited from <a href="#">IFreezable</a> .

**Events**

Name	Description
<a href="#">change</a>	Changes occurred either in the options values, or in the options inheritance hierarchy. Instance of the <a href="#">Event</a> class.
<a href="#">parentchange</a>	The parent object reference changed.  Data fields: <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> Inherited from <a href="#">IChild</a> .

## Methods

Name	Returns	Description
<a href="#">freeze()</a>	<a href="#">IFreezable</a>	Switches the object to "frozen" mode.  Inherited from <a href="#">IFreezable</a> .
<a href="#">get(key[, defaultValue])</a>		Returns the value of the specified option in the context of the existing options inheritance hierarchy. When this method is called, first values are searched for in the current options manager, then, if the value is not defined, the search continues in the hierarchy of parent managers.  Inherited from <a href="#">IOptionManager</a> .
<a href="#">getAll()</a>	Object	Returns a reference to the internal hash that stores option values.  Inherited from <a href="#">IOptionManager</a> .
<a href="#">getName()</a>	String	Returns name of the options manager.  Inherited from <a href="#">IOptionManager</a> .
<a href="#">getNative(key)</a>	Object	Returns the value of the specified option that is defined for the given level of the options hierarchy, i.e. in this manager.  Inherited from <a href="#">IOptionManager</a> .
<a href="#">getParent()</a>	<a href="#">IOptionManager</a>  null	Returns parent options manager.  Inherited from <a href="#">IOptionManager</a> .
<a href="#">isFrozen()</a>	Boolean	Returns true if the object is in "frozen" mode, otherwise false.  Inherited from <a href="#">IFreezable</a> .
<a href="#">resolve(key[, name])</a>	Object	Method intended to be called by child options managers.  Inherited from <a href="#">IOptionManager</a> .

Name	Returns	Description
<code>set(key[, value])</code>	<code>option.Manager</code>	Sets option values for this manager. Two signatures are supported: <ul style="list-style-type: none"> <li>A single argument consisting of a hash in the format {"option name": "option value"}.</li> <li>Two arguments, the first of which is the option name, and the second is the value.</li> </ul>
<code>setName(name)</code>		Sets the name of the options manager. Inherited from <a href="#">IOptionManager</a> .
<code>setParent(parent)</code>	<code>IChild</code>	Sets the parent options manager. Inherited from <a href="#">IOptionManager</a> .
<code>unfreeze()</code>	<code>IFreezable</code>	Switches the object to active mode. Inherited from <a href="#">IFreezable</a> .
<code>unset(keys)</code>	<code>option.Manager</code>	Clears the values for the set options in this manager.
<code>unsetAll()</code>	<code>option.Manager</code>	Clears the values for all options in this manager.

## Events details

### change

Changes occurred either in the options values, or in the options inheritance hierarchy. Instance of the [Event](#) class.

## Methods details

### set

```
{option.Manager} set(key[, value])
```

Sets option values for this manager. Two signatures are supported:

- A single argument consisting of a hash in the format {"option name": "option value"}.
- Two arguments, the first of which is the option name, and the second is the value.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>key</code> *	—	Type: Object String  The option name, or a hash in the format {"option name": "option value"}.
<code>value</code>	—	Type: Object  The option value, if the name was passed as the first argument.

\* Mandatory parameter/option.

### Examples:

#### 1.

```
// Setting multiple options via hash.
myMap.options.set({
  dblClickZoomCentering: true,
  dblClickFloatZoom: true
});
// Generates a single event option.Manager.event:change.
```

#### 2.

```
// Setting options separately.
myMap.options
  .set("dblClickZoomCentering", true)
  .set("dblClickFloatZoom", true);
// Generates the event option.Manager.event:change.
```

#### 3.

```
// Using "freeze" to minimize the number of option.Manager.event:change events.
myMap.options.freeze();
myMap.options.set({
  dblClickZoomCentering: true,
  dblClickFloatZoom: true
});
myMap.options.set('cursor', 'zoom');
myMap.unfreeze();
// Generates a single option.Manager.event:change event.
```

### unset

```
{option.Manager} unset(keys)
```

Clears the values for the set options in this manager.

Returns self-reference.

#### Parameters:

Parameter	Default value	Description
<code>keys</code> *	—	Type: String String[]  Option name or array of option names whose values should be canceled.

\* Mandatory parameter/option.

#### Example:

```
map.options.unset(['dblClickZoomCentering', 'dblClickFloatZoom']);
```

**unsetAll**

```
{option.Manager} unsetAll()
```

Clears the values for all options in this manager.

**Returns** self-reference.

**Example:**

```
var geoObject = new ymaps.Placemark([37, 55], {}, {preset:'islands#blueIcon'});
myMap.geoObjects.add(geoObject);
// Changing the style however we want.
geoObject.options.set({
  iconLayout: 'default#image',
  iconImageHref: 'http://mysite.ru/icon.png',
  iconImageSize: [16, 16]
});
// Restoring the initial appearance.
geoObject.options
  // To avoid a double reaction of the geo object
  // to the option changes, first we call "freeze", then after
  // setting all values, we call "unfreeze".
  .freeze()
  .unsetAll()
  .set('preset','islands#blueIcon')
  .unfreeze();
```

**option.presetStorage**

Static object.

Instance of [util.Storage](#)

Storage for preset options. List of available keys:

**Placemarks with text**

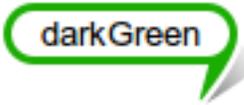
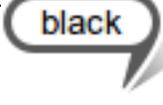
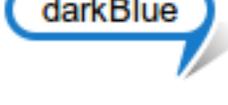
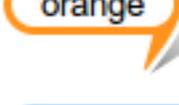
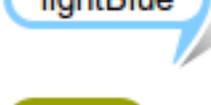
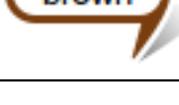
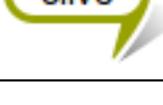
Placemark content is set via [properties.iconContent](#).

Icon	Key	Icon	Key
	'islands#blueIcon'		'islands#darkGreenIcon'
	'islands#redIcon'		'islands#violetIcon'
	'islands#darkOrangeIcon'		'islands#blackIcon'
	'islands#nightIcon'		'islands#yellowIcon'
	'islands#darkBlueIcon'		'islands#greenIcon'
	'islands#pinkIcon'		'islands#orangeIcon'

Icon	Key	Icon	Key
	'islands#grayIcon'		'islands#lightBlueIcon'
	'islands#brownIcon'		'islands#oliveIcon'

### Placemarks with text (icons stretch to fit the content)

Placemark content is set via [properties.iconContent](#).

Icon	Key	Icon	Key
	'islands#blueStretchyIcon'		'islands#darkGreenStretchyIcon'
	'islands#redStretchyIcon'		'islands#violetStretchyIcon'
	'islands#darkOrangeStretchyIcon'		'islands#blackStretchyIcon'
	'islands#nightStretchyIcon'		'islands#yellowStretchyIcon'
	'islands#darkBlueStretchyIcon'		'islands#greenStretchyIcon'
	'islands#pinkStretchyIcon'		'islands#orangeStretchyIcon'
	'islands#grayStretchyIcon'		'islands#lightBlueStretchyIcon'
	'islands#brownStretchyIcon'		'islands#oliveStretchyIcon'

### Placemarks without content with a dot in the center

Icon	Key	Icon	Key
	'islands#blueDotIcon'		'islands#darkGreenDotIcon'

Icon	Key	Icon	Key
	'islands#redDotIcon'		'islands#violetDotIcon'
	'islands#darkOrangeDotIcon'		'islands#blackDotIcon'
	'islands#nightDotIcon'		'islands#yellowDotIcon'
	'islands#darkBlueDotIcon'		'islands#greenDotIcon'
	'islands#pinkDotIcon'		'islands#orangeDotIcon'
	'islands#grayDotIcon'		'islands#lightBlueDotIcon'
	'islands#brownDotIcon'		'islands#oliveDotIcon'

**Placemarks in the style of a circle with text**

Placemark content is set via [properties.iconContent](#).

Icon	Key	Icon	Key
	'islands#blueCircleIcon'		'islands#darkGreenCircleIcon'
	'islands#redCircleIcon'		'islands#violetCircleIcon'
	'islands#darkOrangeCircleIcon'		'islands#blackCircleIcon'
	'islands#nightCircleIcon'		'islands#yellowCircleIcon'
	'islands#darkBlueCircleIcon'		'islands#greenCircleIcon'
	'islands#pinkCircleIcon'		'islands#orangeCircleIcon'
	'islands#grayCircleIcon'		'islands#lightBlueCircleIcon'

Icon	Key	Icon	Key
	'islands#brownCircleIcon'		'islands#oliveCircleIcon'

**Placemarks in the style of circles with a dot in the center**

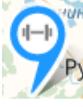
Icon	Key	Icon	Key
	'islands#blueCircleDotIcon'		'islands#darkGreenCircleDotIcon'
	'islands#redCircleDotIcon'		'islands#violetCircleDotIcon'
	'islands#darkOrangeCircleDotIcon'		'islands#blackCircleDotIcon'
	'islands#nightCircleDotIcon'		'islands#yellowCircleDotIcon'
	'islands#darkBlueCircleDotIcon'		'islands#greenCircleDotIcon'
	'islands#pinkCircleDotIcon'		'islands#orangeCircleDotIcon'
	'islands#grayCircleDotIcon'		'islands#lightBlueCircleDotIcon'
	'islands#brownCircleDotIcon'		'islands#oliveCircleDotIcon'

**Placemarks with an icon**

For this type of placemark, the key uses the format: 'islands#{color}{icon}Icon'. For example, 'islands#blueHomeIcon'. The list below shows available icons and corresponding keys. For the list of available colors, see any other table.

Icon	Key	Icon	Key
	'islands#blueAirportIcon'		'islands#blueAttentionIcon'
	'islands#blueAutoIcon'		'islands#blueBarIcon'
	'islands#blueBarberIcon'		'islands#blueBeachIcon'
	'islands#blueBicycleIcon'		'islands#blueBicycle2Icon'

Icon	Key	Icon	Key
	'islands#blueBookIcon'		'islands#blueCarWashIcon'
	'islands#blueChurchIcon'		'islands#blueCinemaIcon'
	'islands#blueCircusIcon'		'islands#blueCourtIcon'
	'islands#blueDeliveryIcon'		'islands#blueDiscountIcon'
	'islands#blueDogIcon'		'islands#blueEducationIcon'
	'islands#blueEntertainmentIcon'		'islands#blueFactoryIcon'
	'islands#blueFamilyIcon'		'islands#blueFashionIcon'
	'islands#blueFoodIcon'		'islands#blueFuelStationIcon'
	'islands#blueGardenIcon'		'islands#blueGovernmentIcon'
	'islands#blueHeartIcon'		'islands#blueHomeIcon'
	'islands#blueHotelIcon'		'islands#blueHydroIcon'
	'islands#blueInfoIcon'		'islands#blueLaundryIcon'
	'islands#blueLeisureIcon'		'islands#blueMassTransitIcon'
	'islands#blueMedicalIcon'		'islands#blueMoneyIcon'

Icon	Key	Icon	Key
	'islands#blueMountainIcon'		'islands#blueNightClubIcon'
	'islands#blueObservatoryIcon'		'islands#blueParkIcon'
	'islands#blueParkingIcon'		'islands#bluePersonIcon'
	'islands#bluePocketIcon'		'islands#bluePoolIcon'
	'islands#bluePostIcon'		'islands#blueRailwayIcon'
	'islands#blueRapidTransitIcon'		'islands#blueRepairShopIcon'
	'islands#blueRunIcon'		'islands#blueScienceIcon'
	'islands#blueShoppingIcon'		'islands#blueSouvenirIcon'
	'islands#blueSportIcon'		'islands#blueStarIcon'
	'islands#blueTheaterIcon'		'islands#blueToiletIcon'
	'islands#blueUnderpassIcon'		'islands#blueVegetationIcon'
	'islands#blueVideoIcon'		'islands#blueWasteIcon'
	'islands#blueWaterParkIcon'		'islands#blueWaterwayIcon'
	'islands#blueWorshipIcon'		'islands#blueZooIcon'

**Placemarks in the style of a circle with an icon**

For this type of placemark, the key uses the format: 'islands#{color}{icon}CircleIcon'. For example, 'islands#blueHomeCircleIcon'. The list below shows available icons and corresponding keys. For the list of available colors, see any other table.

Icon	Key	Icon	Key
	'islands#blueHomeCircleIcon'		'islands#blueScienceCircleIcon'
	'islands#blueAirportCircleIcon'		'islands#blueAttentionCircleIcon'
	'islands#blueAutoCircleIcon'		'islands#blueBarCircleIcon'
	'islands#blueBarberCircleIcon'		'islands#blueBeachCircleIcon'
	'islands#blueBicycleCircleIcon'		'islands#blueBicycle2CircleIcon'
	'islands#blueBookCircleIcon'		'islands#blueCarWashCircleIcon'
	'islands#blueChristianityCircleIcon'		'islands#blueCinemaCircleIcon'
	'islands#blueCircusCircleIcon'		'islands#blueCourtCircleIcon'
	'islands#blueDeliveryCircleIcon'		'islands#blueDiscountCircleIcon'
	'islands#blueDogCircleIcon'		'islands#blueEducationCircleIcon'
	'islands#blueEntertainmentCenterCircleIcon'		'islands#blueFactoryCircleIcon'
	'islands#blueFamilyCircleIcon'		'islands#blueFashionCircleIcon'
	'islands#blueFoodCircleIcon'		'islands#blueFuelStationCircleIcon'
	'islands#blueGardenCircleIcon'		'islands#blueGovernmentCircleIcon'
	'islands#blueHeartCircleIcon'		'islands#blueHomeCircleIcon'
	'islands#blueHotelCircleIcon'		'islands#blueHydroCircleIcon'
	'islands#blueInfoCircleIcon'		'islands#blueLaundryCircleIcon'
	'islands#blueLeisureCircleIcon'		'islands#blueMassTransitCircleIcon'

Icon	Key	Icon	Key
	'islands#blueMedicalCircleIcon'		'islands#blueMoneyCircleIcon'
	'islands#blueMountainCircleIcon'		'islands#blueNightClubCircleIcon'
	'islands#blueObservationCircleIcon'		'islands#blueParkCircleIcon'
	'islands#blueParkingCircleIcon'		'islands#bluePersonCircleIcon'
	'islands#bluePocketCalculatorCircleIcon'		'islands#bluePoolCircleIcon'
	'islands#bluePostCircleIcon'		'islands#blueRailwayCircleIcon'
	'islands#blueRapidTransitCircleIcon'		'islands#blueRepairShopCircleIcon'
	'islands#blueRunCircleIcon'		'islands#blueScienceCircleIcon'
	'islands#blueShoppingCircleIcon'		'islands#blueSouvenirsCircleIcon'
	'islands#blueSportCircleIcon'		'islands#blueStarCircleIcon'
	'islands#blueTheaterCircleIcon'		'islands#blueToiletCircleIcon'
	'islands#blueUnderpassCircleIcon'		'islands#blueVegetationCircleIcon'
	'islands#blueVideoCameraCircleIcon'		'islands#blueWasteCircleIcon'
	'islands#blueWaterParkCircleIcon'		'islands#blueWaterwayCircleIcon'
	'islands#blueWorshipCircleIcon'		'islands#blueZooCircleIcon'

**Pictograms**

Icon	Key
	'islands#geolocationIcon'

**Cluster icons**

Icon	Key	Icon	Key
	'islands#blueClusterIcons'		'islands#invertedBlueClusterIcons'
	'islands#redClusterIcons'		'islands#invertedRedClusterIcons'
	'islands#darkOrangeClusterIcons'		'islands#invertedDarkOrangeClusterIcons'
	'islands#nightClusterIcons'		'islands#invertedNightClusterIcons'
	'islands#darkBlueClusterIcons'		'islands#invertedDarkBlueClusterIcons'
	'islands#pinkClusterIcons'		'islands#invertedPinkClusterIcons'
	'islands#grayClusterIcons'		'islands#invertedGrayClusterIcons'
	'islands#brownClusterIcons'		'islands#invertedBrownClusterIcons'
	'islands#darkGreenClusterIcons'		'islands#invertedDarkGreenClusterIcons'

Icon	Key	Icon	Key
	'islands#violetClusterIcons'		'islands#invertedVioletClusterIcons'
	'islands#blackClusterIcons'		'islands#invertedBlackClusterIcons'
	'islands#yellowClusterIcons'		'islands#invertedYellowClusterIcons'
	'islands#greenClusterIcons'		'islands#invertedGreenClusterIcons'
	'islands#orangeClusterIcons'		'islands#invertedOrangeClusterIcons'
	'islands#lightBlueClusterIcons'		'islands#invertedLightBlueClusterIcons'
	'islands#oliveClusterIcons'		'islands#invertedOliveClusterIcons'

## Methods

### Methods

Name	Returns	Description
<a href="#">add(key, object)</a>	<a href="#">util.Storage</a>	Adds an object to storage.
<a href="#">get(key)</a>	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
<a href="#">remove(key)</a>	<a href="#">util.Storage</a>	Deletes the "key: value" pair from storage.

## overlay

### overlay.Circle

Extends [IOverlay](#).

Interactive overlay for a circle. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use [overlay.storage](#).

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
overlay.Circle(geometry[, data[, options]])
```

#### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IPixelCircleGeometry</a> Pixel geometry of a shape.
<a href="#">data</a>	—	Type: Object Data.
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.fill</a>	—	Type: Boolean Whether there is fill <a href="#">graphics.style.color</a> .
<a href="#">options.fillColor</a>	—	Type: String Fill color.
<a href="#">options.fillImageHref</a>	—	Type: String Background image. When this option is enabled, the "fillColor" value is ignored.
<a href="#">options.fillMethod</a>	'stretch'	Type: String Type of background fill. Accepts one of two values: <ul style="list-style-type: none"> <li>stretch - The background image stretches to fit the size of the overlay.</li> <li>tile - The background image repeats without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.</li> </ul>
<a href="#">options.fillOpacity</a>	—	Type: Number Fill transparency.

Parameter	Default value	Description
<a href="#">options.interactive</a>	true	Type: Boolean  Disables the object's reaction to DOM events.
<a href="#">options.opacity</a>	—	Type: Number  Overall transparency.
<a href="#">options.outline</a>	—	Type: Boolean  Whether there is an outline.
<a href="#">options.separateContainer</a>	—	Type: Boolean  It is drawn on a separate layer.
<a href="#">options.strokeColor</a>	—	Type: String  Line color <a href="#">graphics.style.color</a> .
<a href="#">options.strokeOpacity</a>	—	Type: Number  Contour transparency.
<a href="#">options.strokeStyle</a>	—	Type: Number[] String  Contour style (not supported in Canvas mode) <a href="#">graphics.style.stroke</a> .
<a href="#">options.strokeWidth</a>	—	Type: Number  Line width.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager.  Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">datachange</a>	Data change. Data fields: <ul style="list-style-type: none"><li>• <code>oldData</code> - Old data.</li><li>• <code>newData</code> - New data.</li></ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">emptinesschange</a>	Change to the empty overlay flag. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">IOverlay</a> .
<a href="#">geometrychange</a>	Changed geometry. Data fields: <ul style="list-style-type: none"><li>• <code>oldGeometry</code> - Old pixel geometry.</li><li>• <code>newGeometry</code> - New pixel geometry.</li></ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"><li>• <code>oldMap</code> - Old map.</li><li>• <code>newMap</code> - New map.</li></ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseleave</a>	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mousemove</a>	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseup</a>	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">multitouchend</a>	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface. Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the overlay. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getData()</a>	Object	<p>Returns the overlay data object.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	<p>Returns the current pixel geometry.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getMap()</a>	<a href="#">Map</a>  null	<p>Returns reference to the current map.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>

Name	Returns	Description
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape.  Inherited from <a href="#">IOverlay</a> .
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.  Inherited from <a href="#">IOverlay</a> .
<a href="#">setData(data)</a>		Sets the overlay data.  Inherited from <a href="#">IOverlay</a> .
<a href="#">setGeometry(geometry)</a>		Sets the overlay pixel geometry.  Inherited from <a href="#">IOverlay</a> .
<a href="#">setMap(map)</a>		Sets the map on which to display the overlay.  Inherited from <a href="#">IOverlay</a> .

## overlay.hotspot

### overlay.hotspot.Circle

Extends [IOverlay](#).

Round hotspot overlay. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use [overlay.storage](#).

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
overlay.hotspot.Circle(geometry[, data[, options]])
```

#### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IPixelCircleGeometry</a>  Geometry.
<a href="#">data</a>	—	Type: Object  Data.
<a href="#">options</a>	—	Type: Object  Options.

Parameter	Default value	Description
<a href="#">options.fill</a>	—	Type: Boolean  Whether there is fill <a href="#">graphics.style.color</a> .
<a href="#">options.fillColor</a>	—	Type: String  Fill color.
<a href="#">options.fillImageHref</a>	—	Type: String  Background image. When this option is enabled, the "fillColor" value is ignored.
<a href="#">options.fillMethod</a>	'stretch'	Type: String  Type of background fill. Accepts one of two values: <ul style="list-style-type: none"> <li>stretch - The background image stretches to fit the size of the overlay.</li> <li>tile - The background image repeats without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.</li> </ul>
<a href="#">options.fillOpacity</a>	—	Type: Number  Fill transparency.
<a href="#">options.interactive</a>	true	Type: Boolean  Disables the object's reaction to DOM events.
<a href="#">options.opacity</a>	—	Type: Number  Overall transparency.
<a href="#">options.outline</a>	—	Type: Boolean  Whether there is an outline.
<a href="#">options.separateContainer</a>	—	Type: Boolean  It is drawn on a separate layer.
<a href="#">options.strokeColor</a>	—	Type: String  Line color <a href="#">graphics.style.color</a> .
<a href="#">options.strokeOpacity</a>	—	Type: Number  Contour transparency.
<a href="#">options.strokeStyle</a>	—	Type: Number[] String  Contour style (not supported in Canvas mode) <a href="#">graphics.style.stroke</a> .

Parameter	Default value	Description
<a href="#">options.strokeWidth</a>	—	Type: Number Line width.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">datachange</a>	Data change. Data fields: <ul style="list-style-type: none"> <li>oldData - Old data.</li> <li>newData - New data.</li> </ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">emptinesschange</a>	Change to the empty overlay flag. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">IOverlay</a> .
<a href="#">geometrychange</a>	Changed geometry. Data fields: <ul style="list-style-type: none"> <li>oldGeometry - Old pixel geometry.</li> <li>newGeometry - New pixel geometry.</li> </ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> Inherited from <a href="#">IOverlay</a> .

Name	Description
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>

Name	Description
<a href="#">shapechange</a>	Change to the shape of the area spanning the overlay. Instance of the <a href="#">Event</a> class.  Inherited from <a href="#">IOverlay</a> .
<a href="#">wheel</a>	Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .

## Methods

Name	Returns	Description
<a href="#">getData()</a>	Object	Returns the overlay data object.  Inherited from <a href="#">IOverlay</a> .
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	Returns the current pixel geometry.  Inherited from <a href="#">IOverlay</a> .
<a href="#">getMap()</a>	<a href="#">Map</a>  null	Returns reference to the current map.  Inherited from <a href="#">IOverlay</a> .
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape.  Inherited from <a href="#">IOverlay</a> .
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.  Inherited from <a href="#">IOverlay</a> .
<a href="#">setData(data)</a>		Sets the overlay data.  Inherited from <a href="#">IOverlay</a> .
<a href="#">setGeometry(geometry)</a>		Sets the overlay pixel geometry.  Inherited from <a href="#">IOverlay</a> .
<a href="#">setMap(map)</a>		Sets the map on which to display the overlay.  Inherited from <a href="#">IOverlay</a> .

### **overlay.hotspot.Placemark**

Extends [IOverlay](#).

Point hotspot overlay. By default, the overlays have not been added to `package.full` (the standard set of modules). To create your own overlay instance, use [overlay.storage](#).

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

## Constructor

```
overlay.hotspot.Placemark(geometry[, data[, options]])
```

### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IPixelPointGeometry</a> Pixel geometry of a shape.
<a href="#">data</a>	—	Type: Object Data.
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.cursor</a>	—	Type: String Cursor when the mouse is hovering.
<a href="#">options.interactive</a>	true	Type: Boolean Disables the object's reaction to DOM events.
<a href="#">options.interactivityModel</a>	'default#geoObject'	Type: String Interactivity model. Available keys and their values are listed in the description of <a href="#">interactivityModel.storage</a> .
<a href="#">options.layout</a>	—	Type: Function String Layout. (Type: constructor for an object with the <a href="#">ILayout</a> interface).
<a href="#">options.offset</a>	[0,0]	Type: Array Offset in pixels.
<a href="#">options.pane</a>	'places'	Type: String Container where the placemark layout will be placed.
<a href="#">options.shadow</a>	false	Type: Boolean Flag for whether there is a shadow.
<a href="#">options.shadowLayout</a>	—	Type: Function String Layout for the shadow. (Type: constructor for an object with the <a href="#">ILayout</a> interface).
<a href="#">options.shadowOffset</a>	[0,0]	Type: Array Shadow offset in pixels.

Parameter	Default value	Description
<a href="#">options.shadowsPane</a>	'shadows'	Type: Array  Container where the placemark shadow layout will be placed.
<a href="#">options.zIndex</a>	—	Type: Number  The z-index of the element.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">datachange</a>	Data change. Data fields: <ul style="list-style-type: none"> <li>• <code>oldData</code> - Old data.</li> <li>• <code>newData</code> - New data.</li> </ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">emptinesschange</a>	Change to the empty overlay flag. Instance of the <a href="#">Event</a> class.  Inherited from <a href="#">IOverlay</a> .
<a href="#">geometrychange</a>	Changed geometry. Data fields: <ul style="list-style-type: none"> <li>• <code>oldGeometry</code> - Old pixel geometry.</li> <li>• <code>newGeometry</code> - New pixel geometry.</li> </ul> Inherited from <a href="#">IOverlay</a> .

Name	Description
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldMap</code> - Old map.</li><li>• <code>newMap</code> - New map.</li></ul> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the overlay. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getData()</a>	Object	<p>Returns the overlay data object.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	<p>Returns the current pixel geometry.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getMap()</a>	<a href="#">Map</a>  null	<p>Returns reference to the current map.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	<p>Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">isEmpty()</a>	Boolean	<p>Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>

Name	Returns	Description
<a href="#">setData(data)</a>		Sets the overlay data. Inherited from <a href="#">IOverlay</a> .
<a href="#">setGeometry(geometry)</a>		Sets the overlay pixel geometry. Inherited from <a href="#">IOverlay</a> .
<a href="#">setMap(map)</a>		Sets the map on which to display the overlay. Inherited from <a href="#">IOverlay</a> .

### overlay.hotspot.Polygon

Extends [IOverlay](#).

Polygon hotspot overlay. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use [overlay.storage](#).

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
overlay.hotspot.Polygon(geometry[, data[, options]])
```

#### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IPixelPolygonGeometry</a> The geometry of the shape.
<a href="#">data</a>	—	Type: Object Data.
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.fill</a>	—	Type: String Whether there is fill.
<a href="#">options.fillColor</a>	—	Type: String Fill color <a href="#">graphics.style.color</a> .
<a href="#">options.fillImageHref</a>	—	Type: String Background image. When this option is enabled, the "fillColor" value is ignored.

Parameter	Default value	Description
<a href="#">options.fillMethod</a>	'stretch'	Type: String  Type of background fill. Accepts one of two values: <ul style="list-style-type: none"> <li>stretch - The background image stretches to fit the size of the overlay.</li> <li>tile - The background image is repeated, without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.</li> </ul>
<a href="#">options.fillOpacity</a>	—	Type: Number  Fill transparency.
<a href="#">options.interactive</a>	true	Type: Boolean  Disables the object's reaction to DOM events.
<a href="#">options.opacity</a>	—	Type: Number  Overall transparency.
<a href="#">options.outline</a>	—	Type: String  Whether there is an outline.
<a href="#">options.separateContainer</a>	—	Type: Boolean  It is drawn on a separate layer.
<a href="#">options.strokeColor</a>	—	Type: String  Line color <a href="#">graphics.style.color</a> .
<a href="#">options.strokeOpacity</a>	—	Type: Number  Contour transparency.
<a href="#">options.strokeStyle</a>	—	Type: Number[] String  Contour style <a href="#">graphics.style.stroke</a> .
<a href="#">options.strokeWidth</a>	—	Type: Number  Line width.

\* Mandatory parameter/option.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IDomEventEmitter</a> .

Name	Type	Description
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">datachange</a>	Data change. Data fields: <ul style="list-style-type: none"> <li>• <code>oldData</code> - Old data.</li> <li>• <code>newData</code> - New data.</li> </ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">emptinesschange</a>	Change to the empty overlay flag. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">IOverlay</a> .
<a href="#">geometrychange</a>	Changed geometry. Data fields: <ul style="list-style-type: none"> <li>• <code>oldGeometry</code> - Old pixel geometry.</li> <li>• <code>newGeometry</code> - New pixel geometry.</li> </ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"> <li>• <code>oldMap</code> - Old map.</li> <li>• <code>newMap</code> - New map.</li> </ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseleave</a>	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the overlay. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">getData()</a>	Object	Returns the overlay data object.  Inherited from <a href="#">IOverlay</a> .
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	Returns the current pixel geometry.  Inherited from <a href="#">IOverlay</a> .
<a href="#">getMap()</a>	<a href="#">Map</a>  null	Returns reference to the current map.  Inherited from <a href="#">IOverlay</a> .
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape.  Inherited from <a href="#">IOverlay</a> .
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.  Inherited from <a href="#">IOverlay</a> .
<a href="#">setData(data)</a>		Sets the overlay data.  Inherited from <a href="#">IOverlay</a> .
<a href="#">setGeometry(geometry)</a>		Sets the overlay pixel geometry.  Inherited from <a href="#">IOverlay</a> .
<a href="#">setMap(map)</a>		Sets the map on which to display the overlay.  Inherited from <a href="#">IOverlay</a> .

**overlay.hotspot.Polyline**

Extends [IOverlay](#).

Polyline hotspot overlay. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use [overlay.storage](#).

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
overlay.hotspot.Polyline(geometry[, data[, options]])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IPixelLineStringGeometry</a> Pixel geometry of a shape.
<a href="#">data</a>	—	Type: Object Data.
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.interactive</a>	true	Type: Boolean Disables the object's reaction to DOM events.
<a href="#">options.opacity</a>	—	Type: Number Overall transparency.
<a href="#">options.separateContainer</a>	—	Type: Boolean It is drawn on a separate layer.
<a href="#">options.strokeColor</a>	—	Type: String Line color <a href="#">graphics.style.color</a> .
<a href="#">options.strokeOpacity</a>	—	Type: Number Contour transparency.
<a href="#">options.strokeStyle</a>	—	Type: Number[] String Contour style <a href="#">graphics.style.stroke</a> .
<a href="#">options.strokeWidth</a>	—	Type: Number Line width.

\* Mandatory parameter/option.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">datachange</a>	<p>Data change. Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldData</code> - Old data.</li><li>• <code>newData</code> - New data.</li></ul> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">emptinesschange</a>	<p>Change to the empty overlay flag. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">geometrychange</a>	<p>Changed geometry. Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldGeometry</code> - Old pixel geometry.</li><li>• <code>newGeometry</code> - New pixel geometry.</li></ul> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldMap</code> - Old map.</li><li>• <code>newMap</code> - New map.</li></ul> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the overlay. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getData()</a>	Object	<p>Returns the overlay data object.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>

Name	Returns	Description
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	Returns the current pixel geometry. Inherited from <a href="#">IOverlay</a> .
<a href="#">getMap()</a>	<a href="#">Map</a>  null	Returns reference to the current map. Inherited from <a href="#">IOverlay</a> .
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape. Inherited from <a href="#">IOverlay</a> .
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from <a href="#">IOverlay</a> .
<a href="#">setData(data)</a>		Sets the overlay data. Inherited from <a href="#">IOverlay</a> .
<a href="#">setGeometry(geometry)</a>		Sets the overlay pixel geometry. Inherited from <a href="#">IOverlay</a> .
<a href="#">setMap(map)</a>		Sets the map on which to display the overlay. Inherited from <a href="#">IOverlay</a> .

### overlay.hotspot.Rectangle

Extends [IOverlay](#).

Square hotspot overlay. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use [overlay.storage](#).

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
overlay.hotspot.Rectangle(geometry[, data[, options]])
```

#### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IPixelRectangleGeometry</a> Pixel geometry of a shape.
<a href="#">data</a>	—	Type: Object Data.

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.borderRadius</a>	—	Type: Number Radius of rounded corners.
<a href="#">options.fill</a>	—	Type: String Whether there is fill.
<a href="#">options.fillColor</a>	—	Type: String Fill color <a href="#">graphics.style.color</a> .
<a href="#">options.fillImageHref</a>	—	Type: String Background image. When this option is enabled, the "fillColor" value is ignored.
<a href="#">options.fillMethod</a>	'stretch'	Type: String Type of background fill. Accepts one of two values: <ul style="list-style-type: none"> <li>stretch - The background image stretches to fit the size of the overlay.</li> <li>tile - The background image is repeated, without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.</li> </ul>
<a href="#">options.fillOpacity</a>	—	Type: Number Fill transparency.
<a href="#">options.interactive</a>	true	Type: Boolean Disables the object's reaction to DOM events.
<a href="#">options.opacity</a>	—	Type: Number Overall transparency.
<a href="#">options.outline</a>	—	Type: String Whether there is an outline.
<a href="#">options.separateContainer</a>	—	Type: Boolean It is drawn on a separate layer.
<a href="#">options.strokeColor</a>	—	Type: String Line color <a href="#">graphics.style.color</a> .
<a href="#">options.strokeOpacity</a>	—	Type: Number Contour transparency.

Parameter	Default value	Description
<a href="#">options.strokeStyle</a>	—	Type: Number[] String  Contour style (not supported in Canvas mode) <a href="#">graphics.style.stroke</a> .
<a href="#">options.strokeWidth</a>	—	Type: Number  Line width.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">datachange</a>	Data change. Data fields: <ul style="list-style-type: none"> <li>• <code>oldData</code> - Old data.</li> <li>• <code>newData</code> - New data.</li> </ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">emptinesschange</a>	Change to the empty overlay flag. Instance of the <a href="#">Event</a> class.  Inherited from <a href="#">IOverlay</a> .
<a href="#">geometrychange</a>	Changed geometry. Data fields: <ul style="list-style-type: none"> <li>• <code>oldGeometry</code> - Old pixel geometry.</li> <li>• <code>newGeometry</code> - New pixel geometry.</li> </ul> Inherited from <a href="#">IOverlay</a> .

Name	Description
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldMap</code> - Old map.</li><li>• <code>newMap</code> - New map.</li></ul> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the overlay. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getData()</a>	Object	<p>Returns the overlay data object.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	<p>Returns the current pixel geometry.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getMap()</a>	<a href="#">Map</a>  null	<p>Returns reference to the current map.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	<p>Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">isEmpty()</a>	Boolean	<p>Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>

Name	Returns	Description
<a href="#">setData(data)</a>		Sets the overlay data. Inherited from <a href="#">IOverlay</a> .
<a href="#">setGeometry(geometry)</a>		Sets the overlay pixel geometry. Inherited from <a href="#">IOverlay</a> .
<a href="#">setMap(map)</a>		Sets the map on which to display the overlay. Inherited from <a href="#">IOverlay</a> .

## overlay.html

### overlay.html.Balloon

Extends [IOverlay](#).

HTML overlay for the balloon. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use [overlay.storage](#).

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
overlay.html.Balloon(geometry[, data[, options]])
```

### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IPixelPointGeometry</a> Pixel geometry of a shape.
<a href="#">data</a>	—	Type: Object Data.
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.cursor</a>	—	Type: String Cursor when the mouse is hovering.
<a href="#">options.interactivityModel</a>	"default#opaque"	Type: String Interactivity model. Available keys and their values are listed in the description of <a href="#">interactivityModel.storage</a> .
<a href="#">options.layout</a>	—	Type: Function String Layout. (Type: constructor for an object with the ILayout interface).
<a href="#">options.offset</a>	[0,0]	Type: Array Offset in pixels.

Parameter	Default value	Description
<a href="#">options.pane</a>	"balloon"	Type: String  Container where the balloon layout will be placed.
<a href="#">options.shadow</a>	true	Type: Boolean  Flag for whether there is a shadow.
<a href="#">options.shadowLayout</a>	—	Type: Function String  Shadow layout (type: constructor for an object with the ILayout interface).
<a href="#">options.shadowOffset</a>	[0,0]	Type: Array  Shadow offset in pixels.
<a href="#">options.shadowsPane</a>	"shadows"	Type: Array  Container where the balloon shadow layout will be placed.
<a href="#">options.zIndex</a>	—	Type: Number  The z-index of the element.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager.  Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">datachange</a>	Data change. Data fields: <ul style="list-style-type: none"> <li>oldData - Old data.</li> <li>newData - New data.</li> </ul> Inherited from <a href="#">IOverlay</a> .

Name	Description
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">emptinesschange</a>	<p>Change to the empty overlay flag. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">geometrychange</a>	<p>Changed geometry. Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldGeometry</code> - Old pixel geometry.</li><li>• <code>newGeometry</code> - New pixel geometry.</li></ul> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldMap</code> - Old map.</li><li>• <code>newMap</code> - New map.</li></ul> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the overlay. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getBalloonElement()</a>	HTMLElement	Returns parent element of the balloon layout.
<a href="#">getBalloonLayout()</a>	<a href="#">vow.Promise</a>	Returns Promise object to return the balloon layout.
<a href="#">getBalloonLayoutSync()</a>	<a href="#">ILayout</a>	Returns balloon layout.
<a href="#">getData()</a>	Object	Returns the overlay data object.  Inherited from <a href="#">IOverlay</a> .
<a href="#">getElement()</a>	HTMLElement	Returns parent element of the balloon layout.

Name	Returns	Description
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	Returns the current pixel geometry. Inherited from <a href="#">IOverlay</a> .
<a href="#">getLayout()</a>	<a href="#">vow.Promise</a>	Returns Promise object to return the balloon layout.
<a href="#">getLayoutSync()</a>	<a href="#">ILayout</a>  null	Returns balloon layout.
<a href="#">getMap()</a>	<a href="#">Map</a>  null	Returns reference to the current map. Inherited from <a href="#">IOverlay</a> .
<a href="#">getMode()</a>	String	Returns the current mode of the balloon: "panel" — panel mode, "standard" — standard display.
<a href="#">getShadowElement()</a>	HTMLElement	Returns parent element of the balloon shadow layout.
<a href="#">getShadowLayout()</a>	<a href="#">vow.Promise</a>	Returns Promise object to return the balloon shadow layout.
<a href="#">getShadowLayoutSync()</a>	<a href="#">ILayout</a>  null	Returns balloon shadow layout.
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape. Inherited from <a href="#">IOverlay</a> .
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty or if the layout has not yet been loaded, i.e. it has no content.
<a href="#">setData(data)</a>		Sets the overlay data. Inherited from <a href="#">IOverlay</a> .
<a href="#">setGeometry(geometry)</a>		Sets the overlay pixel geometry. Inherited from <a href="#">IOverlay</a> .
<a href="#">setMap(map)</a>		Sets the map on which to display the overlay. Inherited from <a href="#">IOverlay</a> .

## Methods details

### getBalloonElement

```
{HTMLElement} getBalloonElement()
```

**Returns** parent element of the balloon layout.

### getBalloonLayout

```
{vow.Promise} getBalloonLayout()
```

**Returns** Promise object to return the balloon layout.

### getBalloonLayoutSync

```
{ILayout} getBalloonLayoutSync()
```

**Returns** balloon layout.

### getElement

```
{HTMLElement} getElement()
```

**Returns** parent element of the balloon layout.

### getLayout

```
{vow.Promise} getLayout()
```

**Returns** Promise object to return the balloon layout.

### getLayoutSync

```
{ILayout|null} getLayoutSync()
```

**Returns** balloon layout.

### getMode

```
{String} getMode()
```

**Returns** the current mode of the balloon: "panel" — panel mode, "standard" — standard display.

### getShadowElement

```
{HTMLElement} getShadowElement()
```

**Returns** parent element of the balloon shadow layout.

### getShadowLayout

```
{vow.Promise} getShadowLayout()
```

**Returns** Promise object to return the balloon shadow layout.

### getShadowLayoutSync

```
{ILayout|null} getShadowLayoutSync()
```

**Returns** balloon shadow layout.

### isEmpty

```
{Boolean} isEmpty()
```

**Returns** true if the layout is empty or if the layout has not yet been loaded, i.e. it has no content.

**overlay.html.Hint**

Extends [IOverlay](#).

Simple HTML overlay. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use [overlay.storage](#).

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
overlay.html.Hint(geometry[, data[, options]])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IPixelPointGeometry</a> Pixel geometry of a shape.
<a href="#">data</a>	—	Type: Object Data.
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.cursor</a>	—	Type: String Cursor when the mouse is hovering.
<a href="#">options.interactivityModel</a>	'default#opaque'	Type: String Interactivity model. Available keys and their values are listed in the description of <a href="#">interactivityModel.storage</a> .
<a href="#">options.layout</a>	—	Type: <a href="#">ILayout</a>  String Layout.
<a href="#">options.pane</a>	"outerHint"	Type: String Container where the overlay will be placed.
<a href="#">options.zIndex</a>	—	Type: Number The z-index of the element.

\* Mandatory parameter/option.

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">datachange</a>	<p>Data change. Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldData</code> - Old data.</li> <li>• <code>newData</code> - New data.</li> </ul> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">emptinesschange</a>	<p>Change to the empty overlay flag. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">geometrychange</a>	<p>Changed geometry. Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldGeometry</code> - Old pixel geometry.</li> <li>• <code>newGeometry</code> - New pixel geometry.</li> </ul> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldMap</code> - Old map.</li> <li>• <code>newMap</code> - New map.</li> </ul> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the overlay. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getData()</a>	Object	<p>Returns the overlay data object.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getElement()</a>	HTMLElement	Returns parent element of the icon layout.

Name	Returns	Description
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	Returns the current pixel geometry. Inherited from <a href="#">IOverlay</a> .
<a href="#">getLayout()</a>	<a href="#">vow.Promise</a>	Returns Promise object to return the icon layout.
<a href="#">getLayoutSync()</a>	<a href="#">ILayout</a>  null	Returns icon layout.
<a href="#">getMap()</a>	<a href="#">Map</a>  null	Returns reference to the current map. Inherited from <a href="#">IOverlay</a> .
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape. Inherited from <a href="#">IOverlay</a> .
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty or if the layout has not yet been loaded, i.e. it has no content.
<a href="#">setData(data)</a>		Sets the overlay data. Inherited from <a href="#">IOverlay</a> .
<a href="#">setGeometry(geometry)</a>		Sets the overlay pixel geometry. Inherited from <a href="#">IOverlay</a> .
<a href="#">setMap(map)</a>		Sets the map on which to display the overlay. Inherited from <a href="#">IOverlay</a> .

## Methods details

### getElement

```
{HTMLElement} getElement()
```

Returns parent element of the icon layout.

### getLayout

```
{vow.Promise} getLayout()
```

Returns Promise object to return the icon layout.

### getLayoutSync

```
{ILayout|null} getLayoutSync()
```

Returns icon layout.

## isEmpty

```
{Boolean} isEmpty()
```

**Returns** true if the layout is empty or if the layout has not yet been loaded, i.e. it has no content.

## overlay.html.Placemark

Extends [IOverlay](#).

HTML overlay for the layout. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use [overlay.storage](#).

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

## Constructor

```
overlay.html.Placemark(geometry[, data[, options]])
```

## Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IPixelPointGeometry</a> Pixel geometry.
<a href="#">data</a>	—	Type: Object Data.
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.cursor</a>	—	Type: String Cursor when the mouse is hovering.
<a href="#">options.interactivityModel</a>	'default#geoObject'	Type: String Interactivity model. Available keys and their values are listed in the description of <a href="#">interactivityModel.storage</a> .
<a href="#">options.layout</a>	—	Type: Function String Layout. (Type: constructor for an object with the ILayout interface).
<a href="#">options.offset</a>	[0,0]	Type: Array Offset in pixels.
<a href="#">options.pane</a>	'places'	Type: String Container where the placemark layout will be placed.
<a href="#">options.shadow</a>	false	Type: Boolean Flag for whether there is a shadow.

Parameter	Default value	Description
<a href="#">options.shadowLayout</a>	—	Type: Function String  Layout for the shadow. (Type: constructor for an object with the ILayout interface).
<a href="#">options.shadowOffset</a>	[0,0]	Type: Array  Shadow offset in pixels.
<a href="#">options.shadowsPane</a>	'shadows'	Type: Array  Container where the placemark shadow layout will be placed.
<a href="#">options.zIndex</a>	—	Type: Number  The z-index of the element.

\* Mandatory parameter/option.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

### Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">datachange</a>	Data change. Data fields: <ul style="list-style-type: none"> <li>• <code>oldData</code> - Old data.</li> <li>• <code>newData</code> - New data.</li> </ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">emptinesschange</a>	Change to the empty overlay flag. Instance of the <a href="#">Event</a> class.  Inherited from <a href="#">IOverlay</a> .

Name	Description
<a href="#">geometrychange</a>	<p>Changed geometry. Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldGeometry</code> - Old pixel geometry.</li><li>• <code>newGeometry</code> - New pixel geometry.</li></ul> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldMap</code> - Old map.</li><li>• <code>newMap</code> - New map.</li></ul> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the overlay. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getData()</a>	Object	<p>Returns the overlay data object.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getElement()</a>	HTMLElement	Returns parent element of the icon layout.
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	<p>Returns the current pixel geometry.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getIconElement()</a>	HTMLElement	Returns parent element of the icon layout.
<a href="#">getIconLayout()</a>	<a href="#">vow.Promise</a>	Returns Promise object to return the icon layout.
<a href="#">getIconLayoutSync()</a>	<a href="#">ILayout</a>  null	Returns icon layout.
<a href="#">getLayout()</a>	<a href="#">vow.Promise</a>	Returns Promise object to return the icon layout.
<a href="#">getLayoutSync()</a>	<a href="#">ILayout</a>  null	Returns icon layout.
<a href="#">getMap()</a>	<a href="#">Map</a>  null	<p>Returns reference to the current map.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>

Name	Returns	Description
<a href="#">getShadowElement()</a>	HTMLElement	Returns parent element of the icon shadow layout.
<a href="#">getShadowLayout()</a>	<a href="#">vow.Promise</a>	Returns Promise object to return the icon shadow layout.
<a href="#">getShadowLayoutSync()</a>	<a href="#">ILayout</a>  null	Returns layout for the icon shadow.
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape.  Inherited from <a href="#">IOverlay</a> .
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.  Inherited from <a href="#">IOverlay</a> .
<a href="#">setData(data)</a>		Sets the overlay data.  Inherited from <a href="#">IOverlay</a> .
<a href="#">setGeometry(geometry)</a>		Sets the overlay pixel geometry.  Inherited from <a href="#">IOverlay</a> .
<a href="#">setMap(map)</a>		Sets the map on which to display the overlay.  Inherited from <a href="#">IOverlay</a> .

## Methods details

### getElement

```
{HTMLElement} getElement()
```

Returns parent element of the icon layout.

### getIconElement

```
{HTMLElement} getIconElement()
```

Returns parent element of the icon layout.

### getIconLayout

```
{vow.Promise} getIconLayout()
```

Returns Promise object to return the icon layout.

**getIconLayoutSync**

```
{ILayout|null} getIconLayoutSync()
```

**Returns** icon layout.

**getLayout**

```
{vow.Promise} getLayout()
```

**Returns** Promise object to return the icon layout.

**getLayoutSync**

```
{ILayout|null} getLayoutSync()
```

**Returns** icon layout.

**getShadowElement**

```
{HTMLElement} getShadowElement()
```

**Returns** parent element of the icon shadow layout.

**getShadowLayout**

```
{vow.Promise} getShadowLayout()
```

**Returns** Promise object to return the icon shadow layout.

**getShadowLayoutSync**

```
{ILayout|null} getShadowLayoutSync()
```

**Returns** layout for the icon shadow.

**overlay.html.Rectangle**

Extends [IOverlay](#).

HTML overlay for a rectangle. By default, the overlays have not been added to package.full (the standard set of modules). To create your own overlay instance, use [overlay.storage](#).

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
overlay.html.Rectangle(geometry[, data[, options]])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IPixelRectangleGeometry</a> Pixel geometry of a shape.
<a href="#">data</a>	—	Type: Object Data.

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object  Overlay options.
<a href="#">options.cursor</a>	—	Type: String  Cursor when the mouse is hovering.
<a href="#">options.fillColor</a>	—	Type: String  Fill color <a href="#">graphics.style.color</a> . An option of the standard rectangle layout.
<a href="#">options.fillImageHref</a>	—	Type: String  Background image. When this option is enabled, the "fillColor" value is ignored. An option of the standard rectangle layout.
<a href="#">options.fillMethod</a>	'stretch'	Type: String  Type of background fill. Accepts one of two values: <ul style="list-style-type: none"> <li>stretch - The background image stretches to fit the size of the overlay.</li> <li>tile - The background image is repeated, without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.</li> </ul> An option of the standard rectangle layout.
<a href="#">options.fillOpacity</a>	—	Type: Number  Fill transparency. An option of the standard rectangle layout.
<a href="#">options.interactivityModel</a>	'default#geoObject'	Type: String  Interactivity model. Available keys and their values are listed in the description of <a href="#">interactivityModel.storage</a> .
<a href="#">options.opacity</a>	—	Type: Number  Overall transparency.
<a href="#">options.pane</a>	"areas"	Type: String  Container where the overlay will be placed.
<a href="#">options.strokeColor</a>	—	Type: String  Line color <a href="#">graphics.style.color</a> . An option of the standard rectangle layout.

Parameter	Default value	Description
<a href="#">options.strokeStyle</a>	—	Type: Number[] String  The outline style supported by the standard CSS <i>border-style</i> . An option of the standard rectangle layout.
<a href="#">options.strokeWidth</a>	—	Type: Number  Line width. An option of the standard rectangle layout.
<a href="#">options.zIndex</a>	—	Type: Number  The z-index of the element.
<a href="#">dataSet.options.borderRadius</a>	—	Type: Number  Radius of rounded corners. An option of the standard rectangle layout.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">datachange</a>	Data change. Data fields: <ul style="list-style-type: none"> <li>• <code>oldData</code> - Old data.</li> <li>• <code>newData</code> - New data.</li> </ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> .  Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">emptinesschange</a>	Change to the empty overlay flag. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">IOverlay</a> .
<a href="#">geometrychange</a>	Changed geometry. Data fields: <ul style="list-style-type: none"> <li>oldGeometry - Old pixel geometry.</li> <li>newGeometry - New pixel geometry.</li> </ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseleave</a>	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mousemove</a>	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseup</a>	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">multitouchend</a>	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">multitouchmove</a>	Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the touches property, which contains a list of touches. Every touch is described by an object that contains the following fields: <ul style="list-style-type: none"> <li>clientX - X coordinate of the touch relative to the viewable area of the browser.</li> <li>clientY - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>pageX - X coordinate of the touch relative to the beginning of the document.</li> <li>pageY - Y coordinate of the touch relative to the beginning of the document.</li> </ul> Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the overlay. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getData()</a>	Object	<p>Returns the overlay data object.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getElement()</a>	HTMLElement	<p>Returns parent element of the rectangle layout.</p>
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	<p>Returns the current pixel geometry.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getLayout()</a>	<a href="#">vow.Promise</a>	<p>Returns a promise object to return the rectangle layout.</p>
<a href="#">getLayoutSync()</a>	<a href="#">ILayout</a>  null	<p>Returns rectangle layout.</p>
<a href="#">getMap()</a>	<a href="#">Map</a>  null	<p>Returns reference to the current map.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	<p>Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>

Name	Returns	Description
<code>isEmpty()</code>	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.  Inherited from <a href="#">IOverlay</a> .
<code>setData(data)</code>		Sets the overlay data.  Inherited from <a href="#">IOverlay</a> .
<code>setGeometry(geometry)</code>		Sets the overlay pixel geometry.  Inherited from <a href="#">IOverlay</a> .
<code>setMap(map)</code>		Sets the map on which to display the overlay.  Inherited from <a href="#">IOverlay</a> .

## Methods details

### getElement

```
{HTMLElement} getElement()
```

Returns parent element of the rectangle layout.

### getLayout

```
{vow.Promise} getLayout()
```

Returns a promise object to return the rectangle layout.

### getLayoutSync

```
{ILayout|null} getLayoutSync()
```

Returns rectangle layout.

## overlay.Pin

Extends [IOverlay](#).

Interactive overlay for a circle placemark. By default, the overlays have not been added to `package.full` (the standard set of modules). To create your own overlay instance, use [overlay.storage](#).

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
overlay.Pin(geometry[, data[, options]])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">geometry *</a>	—	Type: <a href="#">IPixelPointGeometry</a>  Pixel geometry of a placemark.
<a href="#">data</a>	—	Type: Object  Data.
<a href="#">options</a>	—	Type: Object  Options.
<a href="#">options.fill</a>	—	Type: Boolean  Whether there is fill <a href="#">graphics.style.color</a> .
<a href="#">options.fillColor</a>	—	Type: String  Fill color.
<a href="#">options.fillImageHref</a>	—	Type: String  Background image. When this option is enabled, the "fillColor" value is ignored.
<a href="#">options.fillMethod</a>	'stretch'	Type: String  Type of background fill. Accepts one of two values: <ul style="list-style-type: none"> <li>stretch - The background image stretches to fit the size of the overlay.</li> <li>tile - The background image repeats without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.</li> </ul>
<a href="#">options.fillOpacity</a>	—	Type: Number  Fill transparency.
<a href="#">options.interactive</a>	true	Type: Boolean  Disables the object's reaction to DOM events.
<a href="#">options.opacity</a>	—	Type: Number  Overall transparency.
<a href="#">options.outline</a>	—	Type: Boolean  Whether there is an outline.
<a href="#">options.radius</a>	—	Type: Number  Radius of the placemark in pixels.
<a href="#">options.separateContainer</a>	—	Type: Boolean  It is drawn on a separate layer.

Parameter	Default value	Description
<a href="#">options.strokeColor</a>	—	Type: String Line color <a href="#">graphics.style.color</a> .
<a href="#">options.strokeOpacity</a>	—	Type: Number Contour transparency.
<a href="#">options.strokeStyle</a>	—	Type: Number[] String Contour style (not supported in Canvas mode) <a href="#">graphics.style.stroke</a> .
<a href="#">options.strokeWidth</a>	—	Type: Number Line width.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">datachange</a>	Data change. Data fields: <ul style="list-style-type: none"> <li>oldData - Old data.</li> <li>newData - New data.</li> </ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">emptinesschange</a>	Change to the empty overlay flag. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">IOverlay</a> .

Name	Description
<a href="#">geometrychange</a>	<p>Changed geometry. Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldGeometry</code> - Old pixel geometry.</li><li>• <code>newGeometry</code> - New pixel geometry.</li></ul> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldMap</code> - Old map.</li><li>• <code>newMap</code> - New map.</li></ul> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the overlay. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getData()</a>	Object	<p>Returns the overlay data object.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	<p>Returns the current pixel geometry.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getMap()</a>	<a href="#">Map</a>  null	<p>Returns reference to the current map.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	<p>Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">isEmpty()</a>	Boolean	<p>Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>

Name	Returns	Description
<a href="#">setData(data)</a>		Sets the overlay data. Inherited from <a href="#">IOverlay</a> .
<a href="#">setGeometry(geometry)</a>		Sets the overlay pixel geometry. Inherited from <a href="#">IOverlay</a> .
<a href="#">setMap(map)</a>		Sets the map on which to display the overlay. Inherited from <a href="#">IOverlay</a> .

## overlay.Placemark

Extends [IOverlay](#).

Placemark overlay. By default, the overlays have not been added to `package.full` (the standard set of modules). To create your own overlay instance, use [overlay.storage](#).

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
overlay.Placemark(geometry[, data[, options]])
```

### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IPixelPointGeometry</a> Pixel geometry.
<a href="#">data</a>	—	Type: Object Data.
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.cursor</a>	—	Type: String Cursor when the mouse is hovering.
<a href="#">options.interactive</a>	true	Type: Boolean Disables the object's reaction to DOM events.
<a href="#">options.interactivityModel</a>	'default#geoObject'	Type: String Interactivity model. Available keys and their values are listed in the description of <a href="#">interactivityModel.storage</a> .
<a href="#">options.layout</a>	—	Type: Function String Layout. (Type: constructor for an object with the <a href="#">ILayout</a> interface).

Parameter	Default value	Description
<a href="#">options.offset</a>	[0,0]	Type: Array Offset in pixels.
<a href="#">options.pane</a>	'places'	Type: String Container where the placemark layout will be placed.
<a href="#">options.shadow</a>	false	Type: Boolean Flag for whether there is a shadow.
<a href="#">options.shadowLayout</a>	—	Type: Function String Layout for the shadow. (Type: constructor for an object with the ILayout interface).
<a href="#">options.shadowOffset</a>	[0,0]	Type: Array Shadow offset in pixels.
<a href="#">options.shadowsPane</a>	'shadows'	Type: Array Container where the placemark shadow layout will be placed.
<a href="#">options.zIndex</a>	—	Type: Number The z-index of the element.

\* Mandatory parameter/option.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

### Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">datachange</a>	Data change. Data fields: <ul style="list-style-type: none"><li>• <code>oldData</code> - Old data.</li><li>• <code>newData</code> - New data.</li></ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">emptinesschange</a>	Change to the empty overlay flag. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">IOverlay</a> .
<a href="#">geometrychange</a>	Changed geometry. Data fields: <ul style="list-style-type: none"><li>• <code>oldGeometry</code> - Old pixel geometry.</li><li>• <code>newGeometry</code> - New pixel geometry.</li></ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"><li>• <code>oldMap</code> - Old map.</li><li>• <code>newMap</code> - New map.</li></ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseleave</a>	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mousemove</a>	Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseup</a>	Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">multitouchend</a>	End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface. Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the overlay. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getData()</a>	Object	<p>Returns the overlay data object.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getElement()</a>	HTMLElement	<p>Returns parent element of the icon layout.</p>
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	<p>Returns the current pixel geometry.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getIconElement()</a>	HTMLElement	<p>Returns parent element of the icon layout.</p>

Name	Returns	Description
<a href="#">getIconLayout()</a>	<a href="#">vow.Promise</a>	Returns Promise object to return the icon layout.
<a href="#">getIconLayoutSync()</a>	<a href="#">ILayout</a>  null	Returns icon layout.
<a href="#">getLayout()</a>	<a href="#">vow.Promise</a>	Returns Promise object to return the icon layout.
<a href="#">getLayoutSync()</a>	<a href="#">ILayout</a>  null	Returns icon layout.
<a href="#">getMap()</a>	<a href="#">Map</a>  null	Returns reference to the current map. Inherited from <a href="#">IOverlay</a> .
<a href="#">getShadowElement()</a>	HTMLElement	Returns parent element of the icon shadow layout.
<a href="#">getShadowLayout()</a>	<a href="#">vow.Promise</a>	Returns Promise object to return the icon shadow layout.
<a href="#">getShadowLayoutSync()</a>	<a href="#">ILayout</a>  null	Returns layout for the icon shadow.
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape. Inherited from <a href="#">IOverlay</a> .
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others. Inherited from <a href="#">IOverlay</a> .
<a href="#">setData(data)</a>		Sets the overlay data. Inherited from <a href="#">IOverlay</a> .
<a href="#">setGeometry(geometry)</a>		Sets the overlay pixel geometry. Inherited from <a href="#">IOverlay</a> .
<a href="#">setMap(map)</a>		Sets the map on which to display the overlay. Inherited from <a href="#">IOverlay</a> .

## Methods details

### getElement

```
{HTMLElement} getElement()
```

**Returns** parent element of the icon layout.

### getIconElement

```
{HTMLElement} getIconElement()
```

**Returns** parent element of the icon layout.

### getIconLayout

```
{vow.Promise} getIconLayout()
```

**Returns** Promise object to return the icon layout.

### getIconLayoutSync

```
{ILayout|null} getIconLayoutSync()
```

**Returns** icon layout.

### getLayout

```
{vow.Promise} getLayout()
```

**Returns** Promise object to return the icon layout.

### getLayoutSync

```
{ILayout|null} getLayoutSync()
```

**Returns** icon layout.

### getShadowElement

```
{HTMLElement} getShadowElement()
```

**Returns** parent element of the icon shadow layout.

### getShadowLayout

```
{vow.Promise} getShadowLayout()
```

**Returns** Promise object to return the icon shadow layout.

### getShadowLayoutSync

```
{ILayout|null} getShadowLayoutSync()
```

**Returns** layout for the icon shadow.

## overlay.Polygon

Extends [IOverlay](#).

Polygon overlay.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
overlay.Polygon(geometry[, data[, options]])
```

**Parameters:**

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IPixelPolygonGeometry</a>  Pixel geometry of a shape.
<a href="#">data</a>	—	Type: Object  Data.
<a href="#">options</a>	—	Type: Object  Options.
<a href="#">options.fill</a>	—	Type: String  Whether there is fill.
<a href="#">options.fillColor</a>	—	Type: String  Fill color <a href="#">graphics.style.color</a> .
<a href="#">options.fillImageHref</a>	—	Type: String  Background image. When this option is enabled, the "fillColor" value is ignored.
<a href="#">options.fillMethod</a>	'stretch'	Type: String  Type of background fill. Accepts one of two values: <ul style="list-style-type: none"> <li>stretch - The background image stretches to fit the size of the overlay.</li> <li>tile - The background image is repeated, without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.</li> </ul>
<a href="#">options.fillOpacity</a>	—	Type: Number  Fill transparency.
<a href="#">options.interactive</a>	true	Type: Boolean  Disables the object's reaction to DOM events.
<a href="#">options.opacity</a>	—	Type: Number  Overall transparency.
<a href="#">options.outline</a>	—	Type: String  Whether there is an outline.
<a href="#">options.separateContainer</a>	—	Type: Boolean  It is drawn on a separate layer.
<a href="#">options.strokeColor</a>	—	Type: String  Line color <a href="#">graphics.style.color</a> .

Parameter	Default value	Description
<a href="#">options.strokeOpacity</a>	—	Type: Number Contour transparency.
<a href="#">options.strokeStyle</a>	—	Type: Number[] String Contour style <a href="#">graphics.style.stroke</a> .
<a href="#">options.strokeWidth</a>	—	Type: Number Line width.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">datachange</a>	Data change. Data fields: <ul style="list-style-type: none"> <li>• <code>oldData</code> - Old data.</li> <li>• <code>newData</code> - New data.</li> </ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">emptinesschange</a>	Change to the empty overlay flag. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">IOverlay</a> .
<a href="#">geometrychange</a>	Changed geometry. Data fields: <ul style="list-style-type: none"> <li>• <code>oldGeometry</code> - Old pixel geometry.</li> <li>• <code>newGeometry</code> - New pixel geometry.</li> </ul> Inherited from <a href="#">IOverlay</a> .

Name	Description
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldMap</code> - Old map.</li><li>• <code>newMap</code> - New map.</li></ul> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the overlay. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getData()</a>	Object	<p>Returns the overlay data object.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	<p>Returns the current pixel geometry.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getMap()</a>	<a href="#">Map</a>  null	<p>Returns reference to the current map.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	<p>Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">isEmpty()</a>	Boolean	<p>Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>

Name	Returns	Description
<a href="#">setData(data)</a>		Sets the overlay data. Inherited from <a href="#">IOverlay</a> .
<a href="#">setGeometry(geometry)</a>		Sets the overlay pixel geometry. Inherited from <a href="#">IOverlay</a> .
<a href="#">setMap(map)</a>		Sets the map on which to display the overlay. Inherited from <a href="#">IOverlay</a> .

## overlay.Polyline

Extends [IOverlay](#).

Line overlay. By default, the overlays have not been added to `package.full` (the standard set of modules). To create your own overlay instance, use [overlay.storage](#).

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
overlay.Polyline(geometry[, data[, options]])
```

### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IPixelLineStringGeometry</a> Pixel geometry of a shape.
<a href="#">data</a>	—	Type: Object Data.
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.interactive</a>	true	Type: Boolean Disables the object's reaction to DOM events.
<a href="#">options.opacity</a>	—	Type: Number Overall transparency.
<a href="#">options.separateContainer</a>	—	Type: Boolean It is drawn on a separate layer.
<a href="#">options.strokeColor</a>	—	Type: String Line color <a href="#">graphics.style.color</a> .
<a href="#">options.strokeOpacity</a>	—	Type: Number Contour transparency.

Parameter	Default value	Description
<a href="#">options.strokeStyle</a>	—	Type: Number[] String Contour style <a href="#">graphics.style.stroke</a> .
<a href="#">options.strokeWidth</a>	—	Type: Number Line width.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">datachange</a>	Data change. Data fields: <ul style="list-style-type: none"> <li>oldData - Old data.</li> <li>newData - New data.</li> </ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">emptinesschange</a>	Change to the empty overlay flag. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">IOverlay</a> .
<a href="#">geometrychange</a>	Changed geometry. Data fields: <ul style="list-style-type: none"> <li>oldGeometry - Old pixel geometry.</li> <li>newGeometry - New pixel geometry.</li> </ul> Inherited from <a href="#">IOverlay</a> .

Name	Description
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldMap</code> - Old map.</li><li>• <code>newMap</code> - New map.</li></ul> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li><code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li><code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li><code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li><code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the overlay. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getData()</a>	Object	<p>Returns the overlay data object.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	<p>Returns the current pixel geometry.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getMap()</a>	<a href="#">Map</a>  null	<p>Returns reference to the current map.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	<p>Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">isEmpty()</a>	Boolean	<p>Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>

Name	Returns	Description
<a href="#">setData(data)</a>		Sets the overlay data. Inherited from <a href="#">IOverlay</a> .
<a href="#">setGeometry(geometry)</a>		Sets the overlay pixel geometry. Inherited from <a href="#">IOverlay</a> .
<a href="#">setMap(map)</a>		Sets the map on which to display the overlay. Inherited from <a href="#">IOverlay</a> .

## overlay.Rectangle

Extends [IOverlay](#).

Polygon overlay. By default, the overlays have not been added to `package.full` (the standard set of modules). To create your own overlay instance, use [overlay.storage](#).

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
overlay.Rectangle(geometry[, data[, options]])
```

### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">IPixelRectangleGeometry</a> Pixel geometry of a shape.
<a href="#">data</a>	—	Type: Object Data.
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.borderRadius</a>	—	Type: Number Radius of rounded corners.
<a href="#">options.fill</a>	—	Type: String Whether there is fill.
<a href="#">options.fillColor</a>	—	Type: String Fill color <a href="#">graphics.style.color</a> .
<a href="#">options.fillImageHref</a>	—	Type: String Background image. When this option is enabled, the "fillColor" value is ignored.

Parameter	Default value	Description
<a href="#">options.fillMethod</a>	'stretch'	Type: String  Type of background fill. Accepts one of two values: <ul style="list-style-type: none"> <li>stretch - The background image stretches to fit the size of the overlay.</li> <li>tile - The background image is repeated, without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.</li> </ul>
<a href="#">options.fillOpacity</a>	—	Type: Number  Fill transparency.
<a href="#">options.interactive</a>	true	Type: Boolean  Disables the object's reaction to DOM events.
<a href="#">options.opacity</a>	—	Type: Number  Overall transparency.
<a href="#">options.outline</a>	—	Type: String  Whether there is an outline.
<a href="#">options.separateContainer</a>	—	Type: Boolean  It is drawn on a separate layer.
<a href="#">options.strokeColor</a>	—	Type: String  Line color <a href="#">graphics.style.color</a> .
<a href="#">options.strokeOpacity</a>	—	Type: Number  Contour transparency.
<a href="#">options.strokeStyle</a>	—	Type: Number[] String  Contour style (not supported in Canvas mode) <a href="#">graphics.style.stroke</a> .
<a href="#">options.strokeWidth</a>	—	Type: Number  Line width.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IDomEventEmitter</a> .

Name	Type	Description
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

## Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">datachange</a>	Data change. Data fields: <ul style="list-style-type: none"> <li>• <code>oldData</code> - Old data.</li> <li>• <code>newData</code> - New data.</li> </ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">emptinesschange</a>	Change to the empty overlay flag. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">IOverlay</a> .
<a href="#">geometrychange</a>	Changed geometry. Data fields: <ul style="list-style-type: none"> <li>• <code>oldGeometry</code> - Old pixel geometry.</li> <li>• <code>newGeometry</code> - New pixel geometry.</li> </ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"> <li>• <code>oldMap</code> - Old map.</li> <li>• <code>newMap</code> - New map.</li> </ul> Inherited from <a href="#">IOverlay</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseleave</a>	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">shapechange</a>	<p>Change to the shape of the area spanning the overlay. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">IOverlay</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">getData()</a>	Object	Returns the overlay data object.  Inherited from <a href="#">IOverlay</a> .
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	Returns the current pixel geometry.  Inherited from <a href="#">IOverlay</a> .
<a href="#">getMap()</a>	<a href="#">Map</a>  null	Returns reference to the current map.  Inherited from <a href="#">IOverlay</a> .
<a href="#">getShape()</a>	<a href="#">IShape</a>  null	Returns a shape that defines the area spanning the overlay in global pixel coordinates, or null if it is not possible to plot the shape.  Inherited from <a href="#">IOverlay</a> .
<a href="#">isEmpty()</a>	Boolean	Returns true if the layout is empty, i.e. it does not have any content. This indicator is used for hiding empty objects such as hints, balloons, and others.  Inherited from <a href="#">IOverlay</a> .
<a href="#">setData(data)</a>		Sets the overlay data.  Inherited from <a href="#">IOverlay</a> .
<a href="#">setGeometry(geometry)</a>		Sets the overlay pixel geometry.  Inherited from <a href="#">IOverlay</a> .
<a href="#">setMap(map)</a>		Sets the map on which to display the overlay.  Inherited from <a href="#">IOverlay</a> .

**overlay.storage**

Static object.

Instance of [util.AsyncStorage](#)

Storage for overlays. By default, the overlays have not been added to `package.full` (the standard set of modules). Overlays are loaded on demand when geo objects are added to the map. To get the overlay class, use the [require](#) method for this storage. By default, the storage declares the following keys for asynchronous access:

- 'default#placemark' - Placemark image overlay [overlay.Placemark](#).
- 'default#pin' - Overlay of the circle [overlay.Pin](#) placemark
- 'default#circle' - Circle overlay [overlay.Circle](#).
- 'default#rectangle' - Rectangle overlay [overlay.Rectangle](#).
- 'default#polyline' - Polyline overlay [overlay.Polyline](#).
- 'default#polygon' - Polygon overlay [overlay.Polygon](#).
- 'hotspot#placemark' - Hotspot placemark overlay [overlay.hotspot.Placemark](#).
- 'hotspot#circle' - Hotspot circle overlay [overlay.hotspot.Circle](#).

- 'hotspot#rectangle' - Hotspot rectangle overlay [overlay.hotspot.Rectangle](#).
- 'hotspot#polyline' - Hotspot polyline overlay [overlay.hotspot.Polyline](#).
- 'hotspot#polygon' - Hotspot polygon overlay [overlay.hotspot.Polygon](#).
- 'html#balloon' - HTML balloon overlay [overlay.html.Balloon](#).
- 'html#hint' - Basic HTML overlay [overlay.html.Hint](#).
- 'html#placemark' - HTML placemark overlay [overlay.html.Placemark](#).
- 'html#rectangle' - HTML rectangle overlay [overlay.html.Rectangle](#).

## Methods

### Example:

```
ymaps.overlay.storage.require(['hotspot#circle'], function (HotspotOverlayClass) {
  // Creating an instance of the received class.
  var overlay = new HotspotOverlayClass(
    new ymaps.geometry.Circle([30, 50], 10), {}, {}
  );
});
```

## Methods

Name	Returns	Description
<a href="#">add(key, object)</a>	<a href="#">util.Storage</a>	Adds an object to storage.
<a href="#">define(key[, depends, resolveCallback[, context]])</a>	<a href="#">util.AsyncStorage</a>	Defines an asynchronous value in storage.
<a href="#">get(key)</a>	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
<a href="#">isDefined(key)</a>	Boolean	Checking if the key can be accessed in the storage.
<a href="#">remove(key)</a>	<a href="#">util.Storage</a>	Deletes the "key: value" pair from storage.
<a href="#">require(keys[, successCallback[, errorCallback[, context]])</a>	<a href="#">vow.Promise</a>	Async request to get values from the storage.

## pane

### pane.EventsPane

Extends [IEventPane](#).

Pane of events.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
pane.EventsPane (map, params)
```

### Parameters:

Parameter	Default value	Description
<a href="#">map</a> *	—	Type: <a href="#">Map</a> Map.

Parameter	Default value	Description
<a href="#">params</a> *	—	Type: Object Parameters.
<a href="#">params.css</a>	—	Type: Object CSS styles of the DOM element of the pane.
<a href="#">params.zIndex</a>	0	Type: Number The zIndex of the pane.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

## Events

Name	Description
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseenter</a>	Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">mouseleave</a>	Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">overflowchange</a>	<p>Change to the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. Instance of <a href="#">IEvent</a>.</p> <p>Inherited from <a href="#">IPane</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">zindexchange</a>	<p>Change to the <code>zIndex</code> value of a pane. Instance of <a href="#">IEvent</a>.</p> <p>Inherited from <a href="#">IPane</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">destroy()</a>		Destroys the pane. Inherited from <a href="#">IPane</a> .
<a href="#">getElement()</a>	HTMLElement	Returns a reference to the pane's DOM container. Inherited from <a href="#">IPane</a> .
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns the map that the pane belongs to. Inherited from <a href="#">IPane</a> .
<a href="#">getOverflow()</a>	String	Returns value of the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. This parameter can take one of the following string values: <ul style="list-style-type: none"> <li>"visible" - When you go off the map container, the content of the pane remains visible.</li> <li>"hidden" - The viewport for the pane content is limited by the map container.</li> </ul> Inherited from <a href="#">IPane</a> .
<a href="#">getZIndex()</a>	Number	Returns the zIndex of the pane. Inherited from <a href="#">IPane</a> .

**pane.MovablePane**

Extends [IContainerPane](#).

Movable map panes.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
pane.MovablePane(map, params)
```

**Parameters:**

Parameter	Default value	Description
<a href="#">map</a> *	—	Type: <a href="#">Map</a> Map.
<a href="#">params</a> *	—	Type: Object Parameters.

Parameter	Default value	Description
<a href="#">params.css</a>	—	Type: Object  CSS styles of the DOM element of the pane.
<a href="#">params.margin</a>	0	Type: Number  Extra padding around the edges of the map container, extending the viewport of the pane.
<a href="#">params.overflow</a>	"hidden"	Type: String  This parameter defines visibility of the pane contents when going outside of the map container. This parameter can take one of the following string values: <ul style="list-style-type: none"> <li>"visible" - When you go off the map container, the content of the pane remains visible.</li> <li>"hidden" - The viewport for the pane content is limited by the map container.</li> </ul>
<a href="#">params.zIndex</a>	0	Type: Number  The zIndex of the pane.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

## Events

Name	Description
<a href="#">actionbegin</a>	The start of pane movement. Instance of <a href="#">IEvent</a> . Inherited from <a href="#">IContainerPane</a> .
<a href="#">actionend</a>	The end of pane movement. Instance of <a href="#">IEvent</a> . Inherited from <a href="#">IContainerPane</a> .
<a href="#">clientpixelschange</a>	Change to the coordinate system of the client pixels. Inherited from <a href="#">IContainerPane</a> .
<a href="#">overflowchange</a>	Change to the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. Instance of <a href="#">IEvent</a> . Inherited from <a href="#">IPane</a> .
<a href="#">viewportchange</a>	Change to a pane's viewport. Instance of <a href="#">IEvent</a> . Inherited from <a href="#">IContainerPane</a> .

Name	Description
<a href="#">zindexchange</a>	Change to the zIndex value of a pane. Instance of <a href="#">IEvent</a> . Inherited from <a href="#">IPane</a> .
<a href="#">zoomchange</a>	Change to the current zoom level of the pane. Instance of <a href="#">IEvent</a> . Inherited from <a href="#">IContainerPane</a> .

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destroys the pane. Inherited from <a href="#">IPane</a> .
<a href="#">fromClientPixels(clientPixelPoint)</a>	Number[]	Converts client pixel coordinates to global coordinates. Inherited from <a href="#">IPositioningContext</a> .
<a href="#">getElement()</a>	HTMLElement	Returns a reference to the pane's DOM container. Inherited from <a href="#">IPane</a> .
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns the map that the pane belongs to. Inherited from <a href="#">IPane</a> .
<a href="#">getOverflow()</a>	String	Returns value of the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. This parameter can take one of the following string values: <ul style="list-style-type: none"> <li>"visible" - When you go off the map container, the content of the pane remains visible.</li> <li>"hidden" - The viewport for the pane content is limited by the map container.</li> </ul> Inherited from <a href="#">IPane</a> .
<a href="#">getViewport()</a>	Number[][]	Returns the viewport for the pane, in client coordinates. Inherited from <a href="#">IContainerPane</a> .
<a href="#">getZIndex()</a>	Number	Returns the zIndex of the pane. Inherited from <a href="#">IPane</a> .

Name	Returns	Description
<a href="#">getZoom()</a>	Number	Returns the current zoom level at which the positioning context works.  Inherited from <a href="#">IPositioningContext</a> .
<a href="#">toClientPixels(globalPixelPoint)</a>	Number[]	Converts global pixel coordinates to client coordinates.  Inherited from <a href="#">IPositioningContext</a> .

## pane.StaticPane

Extends [IContainerPane](#).

A static map pane.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
pane.StaticPane(map, params)
```

### Parameters:

Parameter	Default value	Description
<a href="#">map</a> *	—	Type: <a href="#">Map</a>  Map.
<a href="#">params</a> *	—	Type: Object  Parameters.
<a href="#">params.css</a>	—	Type: Object  CSS styles of the DOM element of the pane.
<a href="#">params.margin</a>	0	Type: Number  Extra padding around the edges of the map container, extending the viewport of the pane.
<a href="#">params.overflow</a>	"hidden"	Type: String  This parameter defines visibility of the pane contents when going outside of the map container. This parameter can take one of the following string values: <ul style="list-style-type: none"> <li>"visible" - When you go off the map container, the content of the pane remains visible.</li> <li>"hidden" - The viewport for the pane content is limited by the map container.</li> </ul>

Parameter	Default value	Description
<a href="#">params.zIndex</a>	0	Type: Number The zIndex of the pane.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

## Events

Name	Description
<a href="#">actionbegin</a>	The start of pane movement. Instance of <a href="#">IEvent</a> . Inherited from <a href="#">IContainerPane</a> .
<a href="#">actionend</a>	The end of pane movement. Instance of <a href="#">IEvent</a> . Inherited from <a href="#">IContainerPane</a> .
<a href="#">clientpixelschange</a>	Change to the coordinate system of the client pixels. Inherited from <a href="#">IContainerPane</a> .
<a href="#">overflowchange</a>	Change to the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. Instance of <a href="#">IEvent</a> . Inherited from <a href="#">IPane</a> .
<a href="#">viewportchange</a>	Change to a pane's viewport. Instance of <a href="#">IEvent</a> . Inherited from <a href="#">IContainerPane</a> .
<a href="#">zindexchange</a>	Change to the zIndex value of a pane. Instance of <a href="#">IEvent</a> . Inherited from <a href="#">IPane</a> .
<a href="#">zoomchange</a>	Change to the current zoom level of the pane. Instance of <a href="#">IEvent</a> . Inherited from <a href="#">IContainerPane</a> .

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destroys the pane. Inherited from <a href="#">IPane</a> .
<a href="#">fromClientPixels(<a href="#">clientPixelPoint</a>)</a>	<a href="#">Number[]</a>	Converts client pixel coordinates to global coordinates. Inherited from <a href="#">IPositioningContext</a> .
<a href="#">getElement()</a>	<a href="#">HTMLElement</a>	Returns a reference to the pane's DOM container. Inherited from <a href="#">IPane</a> .

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns the map that the pane belongs to. Inherited from <a href="#">IPane</a> .
<a href="#">getOverflow()</a>	String	Returns value of the "overflow" parameter, which defines visibility of the pane contents when going outside of the map container. This parameter can take one of the following string values: <ul style="list-style-type: none"> <li>"visible" - When you go off the map container, the content of the pane remains visible.</li> <li>"hidden" - The viewport for the pane content is limited by the map container.</li> </ul> Inherited from <a href="#">IPane</a> .
<a href="#">getViewport()</a>	Number[][]	Returns the viewport for the pane, in client coordinates. Inherited from <a href="#">IContainerPane</a> .
<a href="#">getZIndex()</a>	Number	Returns the zIndex of the pane. Inherited from <a href="#">IPane</a> .
<a href="#">getZoom()</a>	Number	Returns the current zoom level at which the positioning context works. Inherited from <a href="#">IPositioningContext</a> .
<a href="#">toClientPixels(globalPixelPoint)</a>	Number[]	Converts global pixel coordinates to client coordinates. Inherited from <a href="#">IPositioningContext</a> .

## panorama

### panorama.Base

Extends [IPanorama](#).

[Constructor](#) | [Methods](#)

#### Constructor

```
panorama.Base()
```

Initializes the panorama with default parameters.

**Example:**

```
function Panorama () {
  ymaps.panorama.Base.call(this);
  // Making sure that everything is all right with our panorama.
  this.validate();
}

ymaps.util.defineClass(Panorama, ymaps.panorama.Base, {
  getPosition: function () {
    // Let's put our panorama at the center of the coordinate system.
    return [0, 0, 0];
  },

  getCoordSystem: function () {
    return ymaps.coordSystem.cartesian;
  },

  getAngularBBox: function () {
    // We'll make our panorama fully spherical.
    return [0.5 * Math.PI, 2 * Math.PI, -0.5 * Math.PI, 0];
  },

  getTileSize: function () {
    return [512, 512];
  },

  getTileLevels: function () {
    // Our panorama will have just one image.
    return [{
      getTileUrl: function (x, y) {
        return '/' + x + '/' + y + '.jpg';
      },

      getImageSize: function () {
        return [4096, 2048];
      }
    }];
  }
});
```

**Methods**

Name	Returns	Description
<a href="#">getAngularBBox()</a>		Overriding this method is required.
<a href="#">getConnectionArrows()</a>	<a href="#">IPanoramaConnectionArrow[]</a>	Returns an empty array, as if the panorama doesn't have any connection arrows.
<a href="#">getConnectionMarkers()</a>	<a href="#">IPanoramaConnectionMarker[]</a>	Returns an empty array, as if the panorama doesn't have any connection markers.
<a href="#">getConnections()</a>	<a href="#">IPanoramaConnectionMarker[]</a>	 <b>Attention:</b> This method is deprecated.  This method is <b>deprecated</b> . Override <code>panorama.Base.getConnectionMarkers</code> .
<a href="#">getCoordSystem()</a>	<a href="#">ICoordSystem</a>	Returns the geographical coordinate system.
<a href="#">getDefaultDirection()</a>	Number[]	Returns the direction of "north" to the horizon.
<a href="#">getDefaultSpan()</a>	Number[]	Returns the field of view is 130 by 80 degrees in radians.
<a href="#">getGraph()</a>	null	Returns <code>null</code> , as if there aren't any quick transitions for the panorama graph.

Name	Returns	Description
<a href="#">getMarkers()</a>	<a href="#">IPanoramaMarker[]</a>	Returns an empty array, as if the panorama doesn't have any markers.
<a href="#">getName()</a>	String	Returns empty string.
<a href="#">getPosition()</a>		Overriding this method is required.
<a href="#">getThoroughfares()</a>	<a href="#">IPanoramaConnectionArrow[]</a>	<p> <b>Attention:</b> This method is deprecated.</p> <p>This method is <b>deprecated</b>. Override <code>panorama.Base.getConnectionArrows</code>.</p>
<a href="#">getTileLevels()</a>		Overriding this method is required.
<a href="#">getTileSize()</a>		Overriding this method is required.
<a href="#">validate()</a>		<p>Checks the consistency and validity of data returned by methods of the panorama object. Conditions that this method verifies:</p> <ul style="list-style-type: none"> <li>• positions of all objects have three components (including height);</li> <li>• the tile size must be a power of two (for example, 128, 256, or 512 pixels);</li> <li>• the panorama is a full circle (i.e. the width of the angular area must be <math>2\pi</math>);</li> <li>• each zoom level for the panorama image contains an integral number of tiles horizontally (this isn't required vertically).</li> </ul> <p>If calling this method generates an error for the panorama object, we cannot guarantee the stability of the panorama player with this panorama.</p>

## Methods details

### getAngularBBox

```
{ } getAngularBBox()
```

Overriding this method is required.

### getConnectionArrows

```
{IPanoramaConnectionArrow[]} getConnectionArrows()
```

Returns an empty array, as if the panorama doesn't have any connection arrows.

### getConnectionMarkers

```
{IPanoramaConnectionMarker[]} getConnectionMarkers()
```

Returns an empty array, as if the panorama doesn't have any connection markers.

### getConnections

```
{IPanoramaConnectionMarker[]} getConnections()
```

This method is **deprecated**. Override `panorama.Base.getConnectionMarkers`.

**This method is deprecated.**

Returns an empty array, as if the panorama doesn't have any connections.

### getCoordSystem

```
{ICoordSystem} getCoordSystem()
```

Returns the geographical coordinate system.

### getDefaultDirection

```
{Number[]} getDefaultDirection()
```

Returns the direction of "north" to the horizon.

### getDefaultSpan

```
{Number[]} getDefaultSpan()
```

Returns the field of view is 130 by 80 degrees in radians.

### getGraph

```
{null} getGraph()
```

Returns `null`, as if there aren't any quick transitions for the panorama graph.

### getMarkers

```
{IPanoramaMarker[]} getMarkers()
```

Returns an empty array, as if the panorama doesn't have any markers.

### getName

```
{String} getName()
```

Returns empty string.

### getPosition

```
{ } getPosition()
```

Overriding this method is required.

### getThoroughfares

```
{ IPanoramaConnectionArrow[] } getThoroughfares()
```

This method is **deprecated**. Override `panorama.Base.getConnectionArrows`.

**This method is deprecated.**

**Returns** an empty array, as if the panorama doesn't have any transitions.

### getTileLevels

```
{ } getTileLevels()
```

Overriding this method is required.

### getTileSize

```
{ } getTileSize()
```

Overriding this method is required.

### validate

```
{ } validate()
```

Checks the consistency and validity of data returned by methods of the panorama object. Conditions that this method verifies:

- positions of all objects have three components (including height);
- the tile size must be a power of two (for example, 128, 256, or 512 pixels);
- the panorama is a full circle (i.e. the width of the angular area must be  $2\pi$ );
- each zoom level for the panorama image contains an integral number of tiles horizontally (this isn't required vertically).

If calling this method generates an error for the panorama object, we cannot guarantee the stability of the panorama player with this panorama.

### panorama.Base.createPanorama

Static function.

**Returns** panorama instance.

```
{ IPanorama } panorama.Base.createPanorama(params)
```

#### Parameters:

Parameter	Default value	Description
<code>params</code> *	—	Type: Object Panorama parameters.
<code>params.angularBBox</code> *	—	Type: Number[] Angular

Parameter	Default value	Description
<code>params.coordSystem</code>	—	Type: <a href="#">ICoordSystem</a>  Coordinate system that the panorama position is set in. By default, it uses <a href="#">coordSystem.geo</a> .
<code>params.name</code>	"	Type: String  Name of the panorama.
<code>params.position *</code>	—	Type: Number[]  The position of the panorama.
<code>params.tileLevels *</code>	—	Type: <a href="#">IPanoramaTileLevel</a> []  Array of panorama image detail levels.
<code>params.tileSize *</code>	—	Type: Number[]  Size of the panorama image tiles.

\* Mandatory parameter/option.

#### Example:

```
var player = new ymaps.panorama.Player(
  'player',
  ymaps.panorama.Base.createPanorama({
    coordSystem: ymaps.coordSystem.cartesian,
    // Let's put our panorama in the center of the coordinate system.
    position: [0, 0],
    name: 'My panorama',
    // We'll make our panorama fully spherical.
    angularBBox: [0.5 * Math.PI, 2 * Math.PI, -0.5 * Math.PI, 0],
    tileSize: [512, 512],
    tileLevels: [{
      getTileUrl: function (x, y) {
        return '/' + x + '/' + y + '.jpg';
      },
      getImageSize: function () {
        return [4096, 2048];
      }
    }]
  })
);
```

#### `panorama.Base.getMarkerPositionFromDirection`

Static function.

Calculates marker coordinates based on two values: the direction of view to the marker and the distance to it. The coordinates are calculated in the same coordinate system that is used in the panorama.

**Returns** the position of the marker in the coordinate system used in the panorama. Set in the format `[lon, lat, height]`, `[lat, lon, height]` or `[x, y, height]` depending on the coordinate system and order. `height` is the height of the marker in meters, set relative to the same level as the height of the panorama.

```
{ Number[] } panorama.Base.getMarkerPositionFromDirection(panorama, direction, distance)
```

#### Parameters:

Parameter	Default value	Description
<code>panorama *</code>	—	Type: <a href="#">IPanorama</a>  Object describing the panorama.

Parameter	Default value	Description
<a href="#">direction</a> *	—	Type: Number[]  The direction of view to the marker in the <code>[bearing, pitch]</code> format, where: <ul style="list-style-type: none"> <li><code>bearing</code> is the direction to the marker in the horizontal plane; for the geographic coordinate system it represents azimuth (in radians), for the Cartesian - an angle (in radians) starting from the X-axis counterclockwise.</li> <li><code>pitch</code> is the angle of elevation to the marker (in radians).</li> </ul>
<a href="#">distance</a> *	—	Type: Number  Distance to the marker.

\* Mandatory parameter/option.

## panorama.createPlayer

Static function.

Searches for a panorama near the specified point. If at least one is found, it creates a panorama player with this panorama.

**Returns** a promise object that will be resolved by the instance of the class [panorama.Player](#) or rejected with the error description.

```
{ vov.Promise } panorama.createPlayer(element, point[, options])
```

### Parameters:

Parameter	Default value	Description
<a href="#">element</a> *	—	Type: HTMLElement string  A reference to the HTML element that will contain the player, or the ID of this HTML element.
<a href="#">point</a> *	—	Type: Number[]  The point for searching for nearby panoramas.
<a href="#">options</a>	—	Type: Object  Options.

Parameter	Default value	Description
<code>options.direction</code>	'auto'	Type: Number[] String  Direction of view in the format [bearing, pitch], where bearing – is the azimuth of the direction in degrees, pitch – angle of elevation above the horizon in degrees. A special string value <code>auto</code> means that after opening of the panorama, the direction specified in the panorama's metadata will be applied.
<code>options.layer</code>	'yandex#panorama'	Type: String  The layer to search for panoramas. There are two layers available: <ul style="list-style-type: none"> <li>• <code>yandex#panorama</code> - Land panoramas,</li> <li>• <code>yandex#airPanorama</code> - Aerial panoramas.</li> </ul>
<code>options.span</code>	'auto'	Type: Number[] String  The angular dimensions of the field of view in the format [horizontalSpan, verticalSpan], where horizontalSpan – the horizontal field size in degrees, verticalSpan – the vertical field size in degrees.

\* Mandatory parameter/option.

## panorama.isSupported

Static function.

Checks if the panorama player supports the user's platform.

**Returns** `true` if the player supports the browser, otherwise `false`.

```
{ Boolean } panorama.isSupported()
```

## panorama.locate

Static function.

Searches for a panorama at the specified point and layer. The result of the query is an array of found panoramas, represented as [Panorama](#) objects.

**Returns** the promise object that will be resolved by an array of found panoramas (if no panoramas found, the array will be empty) or rejected with the error description.

```
{ vow.Promise } panorama.locate(point[, options])
```

### Parameters:

Parameter	Default value	Description
<code>point</code> *	—	Type: Number[]  The point for searching for nearby panoramas.

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.layer</a>	'yandex#panorama'	Type: String The layer to search for panoramas. There are two layers available: <ul style="list-style-type: none"> <li>• <a href="#">yandex#panorama</a> - Land panoramas,</li> <li>• <a href="#">yandex#airPanorama</a> - Aerial panoramas.</li> </ul>

\* Mandatory parameter/option.

## panorama.Manager

**Note:** The constructor of the `panorama.Manager` class is hidden, as this class is not intended for autonomous initialization.

Extends [IEventEmitter](#).

Manager of the panorama player linked to the map.

[Fields](#) | [Events](#) | [Methods](#)

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .

### Events

Name	Description
<a href="#">closeplayer</a>	The panorama player is closed. Instance of the <a href="#">Event</a> class.
<a href="#">disablelookup</a>	Search mode is disabled for panoramas. Instance of the <a href="#">Event</a> class.
<a href="#">enablelookup</a>	Search mode is enabled for panoramas. Instance of the <a href="#">Event</a> class.
<a href="#">locate</a>	Started searching for panoramas at the given point. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>• <code>point</code> - Search for panoramas around this point.</li> <li>• <code>options</code> - Panorama search options.</li> </ul>
<a href="#">locatefail</a>	The panorama search failed with an error. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>• <code>point</code> - Search for panoramas around this point.</li> <li>• <code>options</code> - Panorama search options.</li> <li>• <code>error</code> - The error that occurred while searching.</li> </ul>

Name	Description
<a href="#">locatesuccess</a>	The panorama search finished successfully. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>point - Search for panoramas around this point.</li> <li>options - Panorama search options.</li> <li>result - The list of the found panoramas.</li> </ul>
<a href="#">openplayer</a>	The panorama player is opened. Instance of the <a href="#">Event</a> class.

## Methods

Name	Returns	Description
<a href="#">closePlayer()</a>		Hides the panorama player.
<a href="#">disableLookup()</a>		Disables panorama search mode on the map.
<a href="#">enableLookup()</a>		Enables panorama search mode on the map.
<a href="#">getPlayer()</a>	<a href="#">panorama.Player</a>	Returns the current panorama player or null if the player is closed.
<a href="#">isLookupEnabled()</a>	Boolean	Checks whether panorama search mode is enabled on the map.
<a href="#">openPlayer(panorama[, locateOptions[, options]])</a>	<a href="#">vow.Promise</a>	Opens the panorama player.

## Events details

### closeplayer

The panorama player is closed. Instance of the [Event](#) class.

### disablelookup

Search mode is disabled for panoramas. Instance of the [Event](#) class.

### enablelookup

Search mode is enabled for panoramas. Instance of the [Event](#) class.

### locate

Started searching for panoramas at the given point. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- point - Search for panoramas around this point.
- options - Panorama search options.

### locatefail

The panorama search failed with an error. Names of fields that are available via the [Event.get](#) method:

- point - Search for panoramas around this point.
- options - Panorama search options.
- error - The error that occurred while searching.

## locatesuccess

The panorama search finished successfully. Names of fields that are available via the [Event.get](#) method:

- point - Search for panoramas around this point.
- options - Panorama search options.
- result - The list of the found panoramas.

## openplayer

The panorama player is opened. Instance of the [Event](#) class.

## Methods details

### closePlayer

```
{ } closePlayer()
```

Hides the panorama player.

### disableLookup

```
{ } disableLookup()
```

Disables panorama search mode on the map.

### enableLookup

```
{ } enableLookup()
```

Enables panorama search mode on the map.

### getPlayer

```
{ panorama.Player } getPlayer()
```

**Returns** the current panorama player or null if the player is closed.

### isLookupEnabled

```
{ Boolean } isLookupEnabled()
```

Checks whether panorama search mode is enabled on the map.

### openPlayer

```
{ vow.Promise } openPlayer(panorama[, locateOptions[, options]])
```

Opens the panorama player.

**Returns** the promise that will be rejected with an error if the panorama has failed to open or if opening has been canceled by the closePlayer request.

### Parameters:

Parameter	Default value	Description
<a href="#">panorama</a> *	—	Type: <a href="#">IPanorama</a>  Number[]  Panorama or the coordinates of the panorama.

Parameter	Default value	Description
<a href="#">locateOptions</a>	—	Type: Object Options for <a href="#">panorama.locate</a> .
<a href="#">options</a>	—	Type: Object
<a href="#">options.direction</a>	'auto'	Type: Number[] Direction of view for panorama.

\* Mandatory parameter/option.

## panorama.Player

Extends [IEventEmitter](#).

Class for creating and controlling the panorama player.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
panorama.Player(element, panorama[, options])
```

Creates an instance of the panorama player.

### Parameters:

Parameter	Default value	Description
<a href="#">element</a> *	—	Type: HTMLElement string A reference to the HTML element that will contain the player, or the ID of this HTML element.
<a href="#">panorama</a> *	—	Type: <a href="#">IPanorama</a> The panorama that will be opened in the created panorama player.
<a href="#">options</a>	—	Type: Object Options.

Parameter	Default value	Description
<a href="#">options.autoFitToViewport</a>	"always"	<p>Type: String</p> <p>The option to disable automatic tracking of the size of the player container. By default, the player always follows the size of its container, and reconstructs the image if the size has changed. Available values:</p> <ul style="list-style-type: none"> <li><code>none</code> — Do not track the size of the container.</li> <li><code>ifNull</code> — As soon as the container gets a CSS "display" value other than "None", the player automatically adjusts its size to the value. After that, tracking stops.</li> <li><code>always</code> — Always follow the size of the container.</li> </ul>
<a href="#">options.controls</a>	—	<p>Type: String[]</p> <p>Set of player controls. Available controls:</p> <ul style="list-style-type: none"> <li><code>closeControl</code> — The button that closes the player.</li> <li><code>fullscreenControl</code> — The button that switches the player to full screen mode.</li> <li><code>panoramaName</code> — The name of the panorama (automatically hidden in the small player).</li> <li><code>zoomControl</code> — Zoom control on the panorama.</li> </ul> <p>The set of controls includes all of the above by default.</p>
<a href="#">options.direction</a>	'auto'	<p>Type: Number[] String</p> <p>Direction of view in the format <code>[bearing, pitch]</code>, where <code>bearing</code> — is the azimuth of the direction in degrees, and <code>pitch</code> is the angle of elevation above the horizon in degrees. The special string value <code>auto</code> means that the direction specified in the panorama's metadata is applied when opening the panorama.</p>
<a href="#">options.hotkeysEnabled</a>	false	<p>Type: Boolean</p> <p>Enables keyboard control of the player. Please note that when you open the player, it begins to intercept some keys (e.g. arrow keys), thus canceling the default reaction of the browser, which may prevent the user from interacting with your page. This is why keyboard control is disabled by default.</p>

Parameter	Default value	Description
<a href="#">options.scrollZoomBehavior</a>	true	Type: Boolean  Disables zooming of the panorama by scrolling. Enabled by default, and the player intercepts the mouse wheel events.
<a href="#">options.span</a>	'auto'	Type: Number[] String  The angular dimensions of the field of view in the format [horizontalSpan, verticalSpan], where horizontalSpan is the horizontal field size in degrees, and verticalSpan is the vertical field size in degrees.
<a href="#">options.suppressMapOpenBlock</a>	false	Type: Boolean  Whether to hide the offer to open the current panorama in Yandex.Maps with all the available map information preserved as completely as possible. true - hide, false - don't hide. The link to Yandex.Maps is displayed in the lower-left corner of the player.

\* Mandatory parameter/option.

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

## Events

Name	Description
<a href="#">destroy</a>	The player was closed by the user or destroyed using the <code>panorama.Player.destroy</code> method.
<a href="#">directionchange</a>	The view direction changed.
<a href="#">error</a>	An error occurred during operation of the player. The user will be shown the appropriate screen.
<a href="#">fullscreenenter</a>	The panorama player switched to full-screen mode.
<a href="#">fullscreenexit</a>	The panorama player exited full-screen mode.
<a href="#">markercollapse</a>	The user clicked on an expanded marker. Names of fields that are available via the <code>Event.get</code> method: <ul style="list-style-type: none"> <li>marker — The marker that was collapsed.</li> </ul>
<a href="#">markerexpand</a>	The user clicked on a collapsed marker. Names of fields that are available via the <code>Event.get</code> method: <ul style="list-style-type: none"> <li>marker — The marker that was expanded.</li> </ul>

Name	Description
<a href="#">markermouseenter</a>	The user's cursor hovered over a marker. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>marker</code> — The marker that the cursor hovered over.</li> </ul>
<a href="#">markermouseleave</a>	The user's cursor left a marker. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>marker</code> — The marker that the cursor moved away from.</li> </ul>
<a href="#">panoramachange</a>	The open panorama changed (for example, as the result of calling the <code>panorama.Player.setPanorama</code> function or a user action).
<a href="#">spanchange</a>	The size of the viewport has been changed.

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		Destroys the player.
<a href="#">fitToViewport()</a>		Checks the size of the player container and if it has changed since the last inspection, rebuilds the image.
<a href="#">getDirection()</a>	Number[]	Returns the current viewing direction, in the format <code>[bearing, pitch]</code> , where <code>bearing</code> is the azimuth of the direction in degrees, and <code>pitch</code> is the angle of elevation above the horizon in degrees.
<a href="#">getPanorama()</a>	<a href="#">IPanorama</a>	Returns the panorama that is currently open in the player.
<a href="#">getSpan()</a>	Number[]	Returns the current angular dimensions of the field of view, in the format <code>[horizontalSpan, verticalSpan]</code> , where <code>horizontalSpan</code> is the horizontal field size in degrees, and <code>verticalSpan</code> is the vertical field size in degrees.
<a href="#">lookAt(point)</a>	<a href="#">panorama.Player</a>	Rotates the view so that the passed point is in the center of the field of view.
<a href="#">moveTo(point[, options])</a>	<a href="#">vow.Promise</a>	Searches for a panorama with the specified parameters and opens it.
<a href="#">setDirection(direction)</a>	<a href="#">panorama.Player</a>	Sets a new viewing direction.
<a href="#">setPanorama(panorama)</a>	<a href="#">panorama.Player</a>	Opens the passed panorama in the player.

Name	Returns	Description
<code>setSpan(<a href="#">span</a>)</code>	<a href="#">panorama.Player</a>	Sets new dimensions for the field of view.

## Events details

### **destroy**

The player was closed by the user or destroyed using the `panorama.Player.destroy` method.

### **directionchange**

The view direction changed.

### **error**

An error occurred during operation of the player. The user will be shown the appropriate screen.

### **fullscreenenter**

The panorama player switched to full-screen mode.

### **fullscreenexit**

The panorama player exited full-screen mode.

### **markercollapse**

The user clicked on an expanded marker. Names of fields that are available via the [Event.get](#) method:

- `marker` — The marker that was collapsed.

### **markerexpand**

The user clicked on a collapsed marker. Names of fields that are available via the [Event.get](#) method:

- `marker` — The marker that was expanded.

### **markermouseenter**

The user's cursor hovered over a marker. Names of fields that are available via the [Event.get](#) method:

- `marker` — The marker that the cursor hovered over.

### **markermouseleave**

The user's cursor left a marker. Names of fields that are available via the [Event.get](#) method:

- `marker` — The marker that the cursor moved away from.

### **panoramachange**

The open panorama changed (for example, as the result of calling the `panorama.Player.setPanorama` function or a user action).

### **spanchange**

The size of the viewport has been changed.

## Methods details

### destroy

```
{ } destroy()
```

Destroys the player.

### fitToViewport

```
{ } fitToViewport()
```

Checks the size of the player container and if it has changed since the last inspection, rebuilds the image.

### getDirection

```
{Number[]} getDirection()
```

**Returns** the current viewing direction, in the format `[bearing, pitch]`, where `bearing` is the azimuth of the direction in degrees, and `pitch` is the angle of elevation above the horizon in degrees.

### getPanorama

```
{IPanorama} getPanorama()
```

**Returns** the panorama that is currently open in the player.

### getSpan

```
{Number[]} getSpan()
```

**Returns** the current angular dimensions of the field of view, in the format `[horizontalSpan, verticalSpan]`, where `horizontalSpan` is the horizontal field size in degrees, and `verticalSpan` is the vertical field size in degrees.

### lookAt

```
{panorama.Player} lookAt(point)
```

Rotates the view so that the passed point is in the center of the field of view.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>point</code> *	—	Type: <code>Number[]</code>  The point to rotate the view at. It can be an array of two or three coordinates. The first two coordinates are interpreted as the geographical coordinates of the point. If three coordinates are passed, the third is interpreted as the height of the point relative to the panorama in meters.

\* Mandatory parameter/option.

**moveTo**

```
{vow.Promise} moveTo(point[, options])
```

Searches for a panorama with the specified parameters and opens it.

**Returns** a promise object that will be resolved if the panorama is found and successfully opened in the player, or rejected with an error message otherwise.

**Parameters:**

Parameter	Default value	Description
<code>point</code> *	—	Type: Number[]  The point for searching for nearby panoramas.
<code>options</code>	—	Type: Object  Options.
<code>options.direction</code>	'auto'	Type: Number[] String  Direction of view in the format <code>[bearing, pitch]</code> , where <code>bearing</code> is the azimuth of the direction in degrees, and <code>pitch</code> is the angle of elevation above the horizon in degrees. The special string value <code>auto</code> means that the direction specified in the panorama's metadata is applied when opening the panorama.
<code>options.layer</code>	'yandex#panorama'	Type: String  The layer to search for panoramas. There are two layers available: <ul style="list-style-type: none"> <li><code>yandex#panorama</code> - Land panoramas.</li> <li><code>yandex#airPanorama</code> - Aerial panoramas.</li> </ul>
<code>options.span</code>	'auto'	Type: Number[] String  The angular dimensions of the field of view in the format <code>[horizontalSpan, verticalSpan]</code> , where <code>horizontalSpan</code> is the horizontal field size, and <code>verticalSpan</code> is the vertical field size.

\* Mandatory parameter/option.

**setDirection**

```
{panorama.Player} setDirection(direction)
```

Sets a new viewing direction.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>direction</code> *	—	Type: <code>Number[] String</code>  Direction of view in the format <code>[bearing, pitch]</code> , where <code>bearing</code> is the azimuth of the direction in degrees, and <code>pitch</code> is the angle of elevation above the horizon in degrees. The special string value <code>auto</code> means that the direction specified in the panorama's metadata is applied when opening the panorama.

\* Mandatory parameter/option.

### setPanorama

```
{panorama.Player} setPanorama(panorama)
```

Opens the passed panorama in the player.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>panorama</code> *	—	Type: <code>IPanorama</code>  Panorama.

\* Mandatory parameter/option.

### setSpan

```
{panorama.Player} setSpan(span)
```

Sets new dimensions for the field of view.

**Returns** self-reference.

**Parameters:**

Parameter	Default value	Description
<code>span</code> *	—	Type: <code>Number[] String</code>  The angular dimensions of the field of view in the format <code>[horizontalSpan, verticalSpan]</code> , where <code>horizontalSpan</code> is the horizontal field size in degrees, and <code>verticalSpan</code> is the vertical field size in degrees.

\* Mandatory parameter/option.

## Panorama

**Note:** The constructor of the Panorama class is hidden, as this class is not intended for autonomous initialization.

Extends [IPanorama](#).

Object describing the panorama.

See [panorama.locate](#)

[Methods](#)

### Methods

Name	Returns	Description
<a href="#">createPlayer</a> ( <code>element</code> [, <code>options</code> ])	<a href="#">vow.Promise</a>	Creates a new instance of the <a href="#">panorama.Player</a> class and opens the panorama in it.
<a href="#">getAngularBBox</a> ()	<code>Number[]</code>	Returns spherical coordinates that define the area covered by the image on the panoramic sphere. Coordinates are specified in the format <code>[thetaTop, phiRight, thetaBottom, phiLeft]</code> (the same as CSS).  Inherited from <a href="#">IPanorama</a> .
<a href="#">getConnectionArrows</a> ()	<a href="#">IPanoramaConnectionArrow</a> []	Returns array of connection arrows on the panorama.  Inherited from <a href="#">IPanorama</a> .
<a href="#">getConnectionMarkers</a> ()	<a href="#">IPanoramaConnectionMarker</a> []	Returns array of connection markers on the panorama.  Inherited from <a href="#">IPanorama</a> .

Name	Returns	Description
<a href="#">getCoordSystem()</a>	<a href="#">ICoordSystem</a>	Returns the coordinate system that is used for defining positions of the panorama and all the associated markers and connections.  Inherited from <a href="#">IPanorama</a> .
<a href="#">getDefaultDirection()</a>	Number[]	Returns default direction of view. It will be used by the player when opening the panorama.  Inherited from <a href="#">IPanorama</a> .
<a href="#">getDefaultSpan()</a>	Number[]	Returns default size of the viewing area. It will be used by the player when opening the panorama.  Inherited from <a href="#">IPanorama</a> .
<a href="#">getGraph()</a>	<a href="#">IPanoramaGraph</a>  null	Returns the graph of panoramas connected to the current panorama for making quick transitions.  Inherited from <a href="#">IPanorama</a> .
<a href="#">getLayer()</a>	String	Returns the layer the panorama belongs to.
<a href="#">getMarkers()</a>	<a href="#">IPanoramaMarker</a> []	Returns array of markers on the panorama.  Inherited from <a href="#">IPanorama</a> .
<a href="#">getName()</a>	String	Returns the name of the panorama displayed by the player in the interface.  Inherited from <a href="#">IPanorama</a> .
<a href="#">getPosition()</a>	Number[]	Returns the location of the panorama in the coordinate system set in the options. Set in the format [lon, lat, height], [lat, lon, height] or [x, y, height] depending on the coordinate system and order. height – the height of the panorama in meters, relative to some level (not necessarily sea level).  Inherited from <a href="#">IPanorama</a> .
<a href="#">getTileLevels()</a>	<a href="#">IPanoramaTileLevel</a> []	Returns array of zoom levels for the panorama image.  Inherited from <a href="#">IPanorama</a> .

Name	Returns	Description
<code>getTileSize()</code>	Number[]	Returns the size of tiles that the panorama image is divided into.  Inherited from <a href="#">IPanorama</a> .

## Methods details

### createPlayer

```
{vow.Promise} createPlayer(element[, options])
```

Creates a new instance of the [panorama.Player](#) class and opens the panorama in it.

**Returns** a promise object that will be resolved by the instance of the class [panorama.Player](#) with the current panorama open.

#### Parameters:

Parameter	Default value	Description
<code>element *</code>	—	Type: HTMLElement string  A reference to the HTML element that will contain the player, or the ID of this HTML element.
<code>options</code>	—	Type: Object  Options.
<code>options.direction</code>	'auto'	Type: Number[] String  Direction of view in the format <code>[bearing, pitch]</code> , where <code>bearing</code> – is the azimuth of the direction in degrees, <code>pitch</code> – angle of elevation above the horizon in degrees. A special string value <code>auto</code> means that after opening of the panorama, the direction specified in the panorama's metadata will be applied.
<code>options.span</code>	'auto'	Type: Number[] String  The angular dimensions of the field of view in the format <code>[horizontalSpan, verticalSpan]</code> , where <code>horizontalSpan</code> – the horizontal field size, <code>verticalSpan</code> – the vertical field size.

\* Mandatory parameter/option.

## getLayer

```
{String} getLayer()
```

See [panorama.locate](#)

Returns the layer the panorama belongs to.

## Placemark

Extends [GeoObject](#).

Placemark. A geo object with the geometry [geometry.Point](#).

See [GeoObject](#) [geometry.Point](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
Placemark(geometry[, properties[, options]])
```

Creates a placemark instance.

### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <a href="#">Number[]</a>   <a href="#">Object</a>   <a href="#">IPointGeometry</a>  Coordinates of the placemark, or a hash describing the geometry, or a reference to the point geometry object.

Parameter	Default value	Description
<a href="#">properties</a>	—	<p>Type: Object <a href="#">IDataManager</a></p> <p>Placemark data. Can be set as a class instance implementing the <a href="#">IDataManager</a> interface, or as a hash. When options are set to default values, the following data fields are interpreted by a geo object:</p> <ul style="list-style-type: none"><li>• <code>iconContent</code> — Content of the geo object's icon.</li><li>• <code>hintContent</code> — Content of the geo object's popup hint.</li><li>• <code>balloonContent</code> — Content of the geo object's balloon.</li><li>• <code>balloonContentHeader</code> — Content of the geo object balloon title.</li><li>• <code>balloonContentBody</code> — Content of the main part of the geo object's balloon.</li><li>• <code>balloonContentFooter</code> — Content of the lower part of the geo object's balloon.</li></ul> <p>The <code>balloonContent</code> field is a shortcut for the <code>balloonContentBody</code> field, but if they are both set simultaneously, <code>balloonContentBody</code> takes priority. You can also add your own custom fields to the placemark data and use them wherever possible. For example, in the placemark layout or balloon layout.</p>

Parameter	Default value	Description
<a href="#">options</a>	—	<p>Type: Object</p> <p>Placemark options.</p> <p><b>To change the style and color of an icon, use the following options:</b></p> <ul style="list-style-type: none"> <li><code>preset</code> — Key for the placemark's preset options. The list of available keys is stored in the <a href="#">option.presetStorage</a> description. You can set the <code>preset</code> option together with the <code>iconColor</code> option only if it takes one of the following values: <ul style="list-style-type: none"> <li>'islands#icon'</li> <li>'islands#dotIcon'</li> <li>'islands#circleIcon'</li> <li>'islands#circleDotIcon'</li> </ul> </li> <li><code>iconColor</code> — The color of the placemark icon. You can specify it in any format that is acceptable in CSS (for example, by name or in the RGB format). This option is used for standard icons in browsers that support SVG. <p><b>Note:</b> This option does not work in IE8.</p> <p><b>If you want to create your own icon layout, you should specify the following options:</b></p> <ul style="list-style-type: none"> <li><code>iconLayout</code> — Icon layout. Type: constructor for an object with the <code>ILayout</code> interface, or its key in the storage. List of available layouts: <ul style="list-style-type: none"> <li>'default#image' — Custom icon image.</li> <li>'default#imageWithContent' — A custom image for an icon with content.</li> </ul> </li> <li>Additional options for the <a href="#">layout.Image</a> and <a href="#">layout.ImageWithContent</a> classes with the "icon" prefix.</li> </ul> <p><b>Note:</b> To specify the layout of the icon shadow, use the same set of options, but with the "iconShadow" prefix. For example, <code>iconShadowLayout</code>.</p> <p><b>You can also set options for individual objects using the Placemark class:</b></p> <ul style="list-style-type: none"> <li>Options for the <a href="#">placemark's balloon</a> with the <code>balloon</code> prefix.</li> <li>Options for the <a href="#">placemark's popup hint</a> with the <code>hint</code> prefix.</li> <li>Options for the polygon's geometry editor with the <code>editor</code> prefix. See the description of the <a href="#">geometryEditor.Point</a> class.</li> <li>Geometry options can be set without a prefix. See the description of the <a href="#">IGeometry</a> class for the</li> </ul> </li> </ul>

Parameter	Default value	Description
<a href="#">options.cursor</a>	"pointer"	Type: String  Type of cursor over a placemark.
<a href="#">options.draggable</a>	false	Type: Boolean  Checks whether the placemark can be dragged.
<a href="#">options.hasBalloon</a>	true	Type: Boolean  Checks whether the placemark has the "balloon" field.
<a href="#">options.hasHint</a>	true	Type: Boolean  Checks whether the placemark has the "hint" field.
<a href="#">options.hideIconOnBalloonOpen</a>	true	Type: Boolean  Hide the placemark when opening the balloon.
<a href="#">options.iconOffset</a>	—	Type: Number[]  The pixel offset of the icon relative to its set position.
<a href="#">options.iconShape</a>	—	Type: <a href="#">IGeometryJson</a>  null  The hotspot shape of the placemark. Specified as a JSON description of the pixel geometry of the icon. Use this option when creating your HTML layouts. The coordinates of the figure geometry are counted from the anchor point.
<a href="#">options.interactiveZIndex</a>	true	Type: Boolean  Enables automatically modifying the z-index of the placemark depending on its state.
<a href="#">options.interactivityModel</a>	"default#geoObject"	Type: String  Interactivity model. Available keys and their values are listed in the description of <a href="#">interactivityModel.storage</a> .
<a href="#">options.openBalloonOnClick</a>	true	Type: Boolean  Checks whether to show the balloon when the placemark is clicked on.
<a href="#">options.openEmptyBalloon</a>	false	Type: Boolean  Checks whether to show an empty balloon when the placemark is clicked on.

Parameter	Default value	Description
<a href="#">options.openEmptyHint</a>	false	Type: Boolean  Checks whether to show an empty hint when the mouse pointer hovers over the placemark.
<a href="#">options.openHintOnHover</a>	true	Type: Boolean  Checks whether to show the hint when the mouse pointer hovers over the placemark.
<a href="#">options.pane</a>	"places"	Type: String  The key of the pane where the placemark overlay is placed.
<a href="#">options.pointOverlay</a>	"default#placemark"	Type: String Function  Key identifier from <a href="#">overlay.storage</a> or the overlay class. The generator function accepts three parameters: <ul style="list-style-type: none"> <li>• geometry: <a href="#">IPixelPointGeometry</a> — The pixel geometry itself.</li> <li>• data: <a href="#">IDataManager</a> or Object — Overlay data.</li> <li>• options: Object — Overlay options.</li> </ul> And returns <a href="#">vow.Promise</a> .
<a href="#">options.syncOverlayInit</a>	false	Type: Boolean  Enables synchronously adding an overlay to the map. By default, overlays are added to the map asynchronously to prevent the browser from hanging when adding a large number of geo objects. However, adding asynchronously does not allow accessing the overlay immediately after adding a placemark to the map.
<a href="#">options.useMapMarginInDragging</a>	true	Type: Boolean  When an object is dragged to the edge of the map, the map center changes automatically. Whether to use map margins when automatically shifting the map center with <a href="#">map.margin.Manager</a> .
<a href="#">options.visible</a>	true	Type: Boolean  Checks placemark visibility.
<a href="#">options.zIndex</a>	—	Type: Number  The z-index of a placemark in its normal state. Lowest priority.

Parameter	Default value	Description
<a href="#">options.zIndexActive</a>	—	Type: Number  The z-index of a placemark icon with an open balloon. Highest priority.
<a href="#">options.zIndexDrag</a>	—	Type: Number  The z-index of a placemark that is being dragged.
<a href="#">options.zIndexHover</a>	—	Type: Number  The z-index of a placemark when the mouse pointer is hovering over it.

\* Mandatory parameter/option.

### Examples:

#### 1.

```
// Creating a placemark.
var placemark = new ymaps.Placemark([55.75, 37.61], {
  balloonContent: '',
  iconContent: "Azerbaijan"
}, {
  preset: "islands#yellowStretchyIcon",
  // Disabling the close balloon button.
  balloonCloseButton: false,
  // The balloon will open and close when the placemark icon is clicked.
  hideIconOnBalloonOpen: false
});
geoMap.geoObjects.add(placemark);
```

#### 2.

```
var placemark = new ymaps.Placemark([55.75, 37.61], {}, {
  // Setting the placemark style (circle).
  preset: "islands#circleDotIcon",
  // Setting the placemark color (in RGB format).
  iconColor: '#ff0000'
});
geoMap.geoObjects.add(placemark);
```

### Fields

Name	Type	Description
<a href="#">balloon</a>	<a href="#">geoObject.Balloon</a>	Balloon for a geo object.  Inherited from <a href="#">GeoObject</a> .
<a href="#">editor</a>	<a href="#">geometryEditor.Point</a>	The "Point" geometry editor.
<a href="#">events</a>	<a href="#">event.Manager</a>	Event manager.  Inherited from <a href="#">GeoObject</a> .
<a href="#">geometry</a>	<a href="#">geometry.Point</a>	The "Point" type of geometry.
<a href="#">hint</a>	<a href="#">geoObject.Hint</a>	Geo object hint.  Inherited from <a href="#">GeoObject</a> .
<a href="#">options</a>	<a href="#">option.Manager</a>	Geo object options manager.  Inherited from <a href="#">GeoObject</a> .

Name	Type	Description
<a href="#">properties</a>	<a href="#">data.Manager</a>	Geo object data manager. Inherited from <a href="#">GeoObject</a> .
<a href="#">state</a>	<a href="#">data.Manager</a>	State of the geo object. Defined by the following fields: <ul style="list-style-type: none"> <li>• active: Boolean - Indicates that a balloon is open on the geo object.</li> <li>• hover: Boolean - Indicates that the mouse is currently pointed at the geo object.</li> <li>• drag: Boolean - Indicates that the geo object is being dragged</li> </ul> Inherited from <a href="#">GeoObject</a> .

## Events

Name	Description
<a href="#">balloonclose</a>	Closing the balloon. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">GeoObject</a> .
<a href="#">balloonopen</a>	Opening a balloon on a geo object. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">GeoObject</a> .
<a href="#">beforedrag</a>	Event preceding the "drag" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>• position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• pixelOffset - Array of two numbers that describe the pixel offset at this step.</li> <li>• domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> Names of methods that are accessible via <a href="#">Event.callMethod</a> : <ul style="list-style-type: none"> <li>• setPixelOffset - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.</li> </ul> If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent drag event will be canceled. Inherited from <a href="#">GeoObject</a> .
<a href="#">beforedragstart</a>	Event preceding the "dragstart" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>• position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> If the <a href="#">Event.preventDefault</a> method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled. Inherited from <a href="#">GeoObject</a> .
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">drag</a>	<p>Dragging a geo object. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>position</code> - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• <code>pixelOffset</code> - Array of two numbers that describe the pixel offset at this step.</li> <li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">dragend</a>	<p>End of geo object dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>position</code> - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">dragstart</a>	<p>Start of geo object dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>position</code> - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">editorstatechange</a>	<p>Change in the state of the editor for the geo object's geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code> - Original event of the geometry editor.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the geometry.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">hintclose</a>	<p>Closing the hint. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">hintopen</a>	<p>Opening a hint on a geo object. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldMap</code> - Old map.</li> <li>• <code>newMap</code> - New map.</li> </ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>

Name	Description
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>overlay: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>oldOverlay: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	<p>Returns reference to the map.</p> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	<p>Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	<p>The method provides synchronous access to the overlay.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	<p>Returns link to the parent object, or null if the parent element was not set.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	<p>Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>

## Fields details

### editor

```
{geometryEditor.Point} editor
```

The "Point" geometry editor.

### geometry

```
{geometry.Point} geometry
```

The "Point" type of geometry.

## Polygon

Extends [GeoObject](#).

Polygon. A geo object with the geometry [geometry.Polygon](#).

**See** [GeoObject](#) [geometry.Polygon](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
Polygon(geometry[, properties[, options]])
```

Creates a polygon instance.

### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	Type: <code>Number[][][] Object IPolygonGeometry</code>  Coordinates of polyline vertexes that define the outer and inner boundaries of the polygon, a hash object with geometry parameters, or a reference to the geometry object. The inner boundary can be omitted.

Parameter	Default value	Description
<a href="#">properties</a>	—	<p>Type: Object <a href="#">IDataManager</a></p> <p>Polygon data. Can be set as a class instance implementing the <a href="#">IDataManager</a> interface, or as a hash. When options are set to default values, the following data fields are interpreted by a geo object:</p> <ul style="list-style-type: none"> <li>hintContent - Content of the polygon's popup hint.</li> <li>balloonContent - Content of the polygon's balloon.</li> <li>balloonContentHeader - Content of the polygon balloon title.</li> <li>balloonContentBody - Content of the main part of the polygon's balloon.</li> <li>balloonContentFooter - Content of the lower part of the polygon's balloon.</li> </ul> <p>The balloonContent field is a shortcut for the balloonContentBody field, but if they are both set simultaneously, balloonContentBody takes priority. You can also add your own custom fields to the polygon data and use them wherever possible. For example, in the polygon balloon layout.</p>
<a href="#">options</a>	—	<p>Type: Object</p> <p>Polygon options. Using this parameter, you can set options for the polygon itself, as well as for its parts:</p> <ul style="list-style-type: none"> <li>Options for the <a href="#">polygon's balloon</a> with the <code>balloon</code> prefix.</li> <li>Options for the <a href="#">polygon's popup hint</a> with the <code>hint</code> prefix.</li> <li>Options for the polygon's geometry editor with the <code>editor</code> prefix. See the description of the <a href="#">geometryEditor.Polygon</a> class.</li> <li>Geometry options can be set without a prefix. See the description of the <a href="#">IGeometry</a> class for the <a href="#">geometry.Polygon</a> geometry.</li> </ul>
<a href="#">options.cursor</a>	"pointer"	<p>Type: String</p> <p>Type of cursor over a polygon.</p>
<a href="#">options.draggable</a>	false	<p>Type: Boolean</p> <p>Checks whether the polygon can be dragged.</p>
<a href="#">options.fill</a>	true	<p>Type: Boolean</p> <p>Whether the shape is filled.</p>

Parameter	Default value	Description
<a href="#">options.fillColor</a>	"0066ff99"	Type: String Fill color.
<a href="#">options.fillImageHref</a>	—	Type: String Background image. When this option is enabled in <b>stretch</b> mode, the "fillColor" value is ignored.
<a href="#">options.fillMethod</a>	'stretch'	Type: String Type of background fill. Accepts one of two values: <ul style="list-style-type: none"> <li>stretch - The background image stretches to fit the size of the overlay.</li> <li>tile - The background image is repeated, without changing its size. This is similar to background-repeat in CSS. It can be used for filling a shape with a template.</li> </ul>
<a href="#">options.fillOpacity</a>	1	Type: Number Fill transparency.
<a href="#">options.hasBalloon</a>	true	Type: Boolean Checks if the polygon has the "balloon" field.
<a href="#">options.hasHint</a>	true	Type: Boolean Checks if the polygon has the "hint" field.
<a href="#">options.interactiveZIndex</a>	false	Type: Boolean Enables to automatically modify z-index of the polygon depending on its state.
<a href="#">options.interactivityModel</a>	"default#geoObject"	Type: String Interactivity model. Available keys and their values are listed in the description of <a href="#">interactivityModel.storage</a> .
<a href="#">options.opacity</a>	1	Type: Number Transparency.
<a href="#">options.openBalloonOnClick</a>	true	Type: Boolean Checks whether to show the balloon when the polygon is clicked on.
<a href="#">options.openEmptyBalloon</a>	false	Type: Boolean Checks whether to show an empty balloon when the polygon is clicked on.

Parameter	Default value	Description
<a href="#">options.openEmptyHint</a>	false	Type: Boolean  Checks whether to show an empty hint when the mouse pointer hovers over the polygon.
<a href="#">options.openHintOnHover</a>	true	Type: Boolean  Checks whether to show an empty hint when the mouse pointer hovers over the polygon.
<a href="#">options.outline</a>	true	Type: Boolean  Whether the shape has an outline.
<a href="#">options.pane</a>	"areas"	Type: String  The key of the pane where the polygon overlay is placed.
<a href="#">options.polygonOverlay</a>	"default#polygon"	Type: String Function  Key identifier from <a href="#">overlay.storage</a> or the overlay class. The generator function accepts three parameters: <ul style="list-style-type: none"> <li>• geometry: <a href="#">IPixelPolygonGeometry</a> - The pixel geometry itself.</li> <li>• data: <a href="#">IDataManager</a> or Object - Overlay data.</li> <li>• options: Object - The overlay options.</li> </ul> And returns <a href="#">vow.Promise</a> .
<a href="#">options.strokeColor</a>	"0066ffff"	Type: String String[]  Color of the line or outline. You can set multiple values for a multistroke outline.
<a href="#">options.strokeOpacity</a>	1	Type: Number Number[]  Transparency of the line or outline. You can set multiple values for a multistroke outline.
<a href="#">options.strokeStyle</a>	—	Type: String Object String[] Object[]  Style of the line or outline. You can set multiple values for a multistroke outline.
<a href="#">options.strokeWidth</a>	1	Type: Number Number[]  Thickness of the line or outline. You can set multiple values for a multistroke outline.

Parameter	Default value	Description
<a href="#">options.syncOverlayInit</a>	false	Type: Boolean  Enables synchronously adding an overlay to the map. By default, overlays are added to the map asynchronously to prevent the browser from hanging when adding a large number of geo objects. However, adding asynchronously does not allow accessing the overlay immediately after adding a polygon to the map.
<a href="#">options.useMapMarginInDragging</a>	true	Type: Boolean  When an object is dragged to the edge of the map, the map center changes automatically. Whether to use map margins when automatically shifting the map center with <a href="#">map.margin.Manager</a> .
<a href="#">options.visible</a>	true	Type: Boolean  Checks the visibility of the polygon.
<a href="#">options.zIndex</a>	—	Type: Number  The z-index of the polygon in the normal state. Lowest priority.
<a href="#">options.zIndexActive</a>	—	Type: Number  The z-index of the polygon with an opened balloon. Highest priority.
<a href="#">options.zIndexDrag</a>	—	Type: Number  The z-index of the polygon while dragging.
<a href="#">options.zIndexHover</a>	—	Type: Number  The z-index of a polygon when the mouse pointer is hovering over it.

\* Mandatory parameter/option.

#### Example:

```
var polygon = new ymaps.Polygon([
  // Coordinates of the outer contour.
  [[-80, 60], [-90, 50], [-60, 40], [-80, 60]],
  // Coordinates of the inner contour.
  [[-90, 80], [-90, 30], [-20, 40], [-90, 80]]
], {
  hintContent: "Polygon"
}, {
  fillColor: '#6699ff',
  // Making the polygon transparent for map events.
  interactivityModel: 'default#transparent',
  strokeWidth: 8,
  opacity: 0.5
});
myMap.geoObjects.add(polygon);
myMap.setBounds(polygon.geometry.getBounds());
```

**Fields**

Name	Type	Description
<a href="#">balloon</a>	<a href="#">geoObject.Balloon</a>	Balloon for a geo object. Inherited from <a href="#">GeoObject</a> .
<a href="#">editor</a>	<a href="#">geometryEditor.Polygon</a>	The "Polygon" geometry editor.
<a href="#">events</a>	<a href="#">event.Manager</a>	Event manager. Inherited from <a href="#">GeoObject</a> .
<a href="#">geometry</a>	<a href="#">geometry.Polygon</a>	The "Polygon" type of geometry.
<a href="#">hint</a>	<a href="#">geoObject.Hint</a>	Geo object hint. Inherited from <a href="#">GeoObject</a> .
<a href="#">options</a>	<a href="#">option.Manager</a>	Geo object options manager. Inherited from <a href="#">GeoObject</a> .
<a href="#">properties</a>	<a href="#">data.Manager</a>	Geo object data manager. Inherited from <a href="#">GeoObject</a> .
<a href="#">state</a>	<a href="#">data.Manager</a>	State of the geo object. Defined by the following fields: <ul style="list-style-type: none"> <li>• active: Boolean - Indicates that a balloon is open on the geo object.</li> <li>• hover: Boolean - Indicates that the mouse is currently pointed at the geo object.</li> <li>• drag: Boolean - Indicates that the geo object is being dragged</li> </ul> Inherited from <a href="#">GeoObject</a> .

**Events**

Name	Description
<a href="#">balloonclose</a>	Closing the balloon. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">GeoObject</a> .
<a href="#">balloonopen</a>	Opening a balloon on a geo object. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">GeoObject</a> .
<a href="#">beforedrag</a>	Event preceding the "drag" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>• position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• pixelOffset - Array of two numbers that describe the pixel offset at this step.</li> <li>• domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> Names of methods that are accessible via <a href="#">Event.callMethod</a> : <ul style="list-style-type: none"> <li>• setPixelOffset - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.</li> </ul> If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent drag event will be canceled. Inherited from <a href="#">GeoObject</a> .

Name	Description
<a href="#">beforedragstart</a>	<p>Event preceding the "dragstart" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">drag</a>	<p>Dragging a geo object. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• pixelOffset - Array of two numbers that describe the pixel offset at this step.</li> <li>• domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">dragend</a>	<p>End of geo object dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">dragstart</a>	<p>Start of geo object dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">editorstatechange</a>	<p>Change in the state of the editor for the geo object's geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• originalEvent - Original event of the geometry editor.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>

Name	Description
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li></ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">hintclose</a>	<p>Closing the hint. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">hintopen</a>	<p>Opening a hint on a geo object. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>oldMap - Old map.</li><li>newMap - New map.</li></ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IChildOnMap</a> .

**Fields details****editor**

```
{geometryEditor.Polygon} editor
```

The "Polygon" geometry editor.

**geometry**

```
{geometry.Polygon} geometry
```

The "Polygon" type of geometry.

**Polyline**

Extends [GeoObject](#).

Polyline. A geo object with the geometry [geometry.LineString](#).

**See** [GeoObject](#) [geometry.LineString](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
Polyline(geometry[, properties[, options]])
```

Creates a polyline instance.

**Parameters:**

Parameter	Default value	Description
<a href="#">geometry</a> *	—	<p>Type: <code>Number[][]</code>  <code>Object ILineStringGeometry</code></p> <p>Coordinates of the vertexes, a hash object with geometry parameters, or a reference to the polyline geometry object.</p>
<a href="#">properties</a>	—	<p>Type: <code>Object IDataManager</code></p> <p>Polyline data. Can be set as a class instance implementing the <a href="#">IDataManager</a> interface, or as a hash. When options are set to default values, the following data fields are interpreted by a geo object:</p> <ul style="list-style-type: none"> <li>• <code>hintContent</code> - Content of the polyline's popup hint.</li> <li>• <code>balloonContent</code> - Content of the polyline's balloon.</li> <li>• <code>balloonContentHeader</code> - Content of the polyline balloon title.</li> <li>• <code>balloonContentBody</code> - Content of the main part of the polyline's balloon.</li> <li>• <code>balloonContentFooter</code> - Content of the lower part of the polyline's balloon.</li> </ul> <p>The <code>balloonContent</code> field is a shortcut for the <code>balloonContentBody</code> field, but if they are both set simultaneously, <code>balloonContentBody</code> takes priority. You can also add your own custom fields to the polyline data and use them, for example, in the popup hint layout.</p>
<a href="#">options</a>	—	<p>Type: <code>Object</code></p> <p>Polyline options. Using this parameter, you can set options for the polyline itself, as well as for its parts:</p> <ul style="list-style-type: none"> <li>• Options for the polyline <a href="#">balloon</a> with the <code>balloon</code> prefix.</li> <li>• Options for the polyline's <a href="#">popup</a> hint with the <code>hint</code> prefix.</li> <li>• Options for the polyline's geometry editor with the <code>editor</code> prefix. See the description of the <a href="#">geometryEditor.LineString</a> class.</li> <li>• Geometry options can be set without a prefix. See the description of the <a href="#">IGeometry</a> class for the <a href="#">geometry.LineString</a> geometry.</li> </ul>
<a href="#">options.cursor</a>	"pointer"	<p>Type: <code>String</code></p> <p>Type of cursor over a polyline.</p>

Parameter	Default value	Description
<a href="#">options.draggable</a>	false	Type: Boolean  Checks whether the polyline can be dragged.
<a href="#">options.hasBalloon</a>	true	Type: Boolean  Checks whether the polyline has the "balloon" field.
<a href="#">options.hasHint</a>	true	Type: Boolean  Checks whether the polyline has the "hint" field.
<a href="#">options.interactiveZIndex</a>	false	Type: Boolean  Enables automatically modifying the z-index of the polyline depending on its state.
<a href="#">options.interactivityModel</a>	"default#geoObject"	Type: String  Interactivity model. Available keys and their values are listed in the description of <a href="#">interactivityModel.storage</a> .
<a href="#">options.lineStringOverlay</a>	"default#polyline"	Type: String Function  Key identifier from <a href="#">overlay.storage</a> or the overlay class. The generator function accepts three parameters: <ul style="list-style-type: none"> <li>• geometry: <a href="#">IPixelLineStringGeometry</a> - The pixel geometry itself.</li> <li>• data: <a href="#">IDataManager</a> or Object - Overlay data.</li> <li>• options: Object - The overlay options.</li> </ul> And returns <a href="#">vow.Promise</a> .
<a href="#">options.opacity</a>	1	Type: Number  Transparency.
<a href="#">options.openBalloonOnClick</a>	true	Type: Boolean  Checks whether to show the balloon when the polyline is clicked on.
<a href="#">options.openEmptyBalloon</a>	false	Type: Boolean  Checks whether to show an empty balloon when the polyline is clicked on.
<a href="#">options.openEmptyHint</a>	false	Type: Boolean  Checks whether to show an empty hint when the mouse pointer hovers over the polyline.

Parameter	Default value	Description
<a href="#">options.openHintOnHover</a>	true	Type: Boolean  Checks whether to show an empty hint when the mouse pointer hovers over the polyline.
<a href="#">options.pane</a>	"areas"	Type: <a href="#">IPane</a>  String  The key of the pane where the polyline overlay is placed.
<a href="#">options.strokeColor</a>	"0066ffff"	Type: String String[]  Color of the line or outline. You can set multiple values for a multistroke outline.
<a href="#">options.strokeOpacity</a>	1	Type: Number Number[]  Transparency of the line or outline. You can set multiple values for a multistroke outline.
<a href="#">options.strokeStyle</a>	—	Type: String Object String[] Object[]  Style of the line or outline. You can set multiple values for a multistroke outline.
<a href="#">options.strokeWidth</a>	1	Type: Number Number[]  Thickness of the line or outline. You can set multiple values for a multistroke outline.
<a href="#">options.syncOverlayInit</a>	false	Type: Boolean  Enables synchronously adding an overlay to the map. By default, overlays are added to the map asynchronously to prevent the browser from hanging when adding a large number of polylines. However, adding asynchronously does not allow accessing the overlay immediately after adding a polyline to the map.
<a href="#">options.useMapMarginInDragging</a>	true	Type: Boolean  When an object is dragged to the edge of the map, the map center changes automatically. Whether to use map margins when automatically shifting the map center with <a href="#">map.margin.Manager</a> .
<a href="#">options.visible</a>	true	Type: Boolean  Checks polyline visibility.
<a href="#">options.zIndex</a>	—	Type: Number  The z-index of a polyline in its normal state. Lowest priority.

Parameter	Default value	Description
<a href="#">options.zIndexActive</a>	—	Type: Number  The z-index of a polyline with an open balloon. Highest priority.
<a href="#">options.zIndexDrag</a>	—	Type: Number  The z-index of a polyline that is being dragged.
<a href="#">options.zIndexHover</a>	—	Type: Number  The z-index of a polyline when the mouse pointer is hovering over it.

\* Mandatory parameter/option.

#### Example:

```
// Creating a polyline
var polyline = new ymaps.Polyline([
  [-80, 60], [-90, 50], [-60, 40], [-80, 60]
], {
  hintContent: "Polyline"
}, {
  draggable: true,
  strokeColor: '#000000',
  strokeWidth: 4,
  // The first number sets the stroke length. The second number sets the gap length.
  strokeStyle: '1 5'
});
// Adding the polyline to the map.
myMap.geoObjects.add(polyline);
// Setting the polyline borders for the map.
myMap.setBounds(polyline.geometry.getBounds());
```

#### Fields

Name	Type	Description
<a href="#">balloon</a>	<a href="#">geoObject.Balloon</a>	Balloon for a geo object.  Inherited from <a href="#">GeoObject</a> .
<a href="#">editor</a>	<a href="#">geometryEditor.LineString</a>	The "Polyline" geometry editor.
<a href="#">events</a>	<a href="#">event.Manager</a>	Event manager.  Inherited from <a href="#">GeoObject</a> .
<a href="#">geometry</a>	<a href="#">geometry.LineString</a>	"Polyline" type of geometry.
<a href="#">hint</a>	<a href="#">geoObject.Hint</a>	Geo object hint.  Inherited from <a href="#">GeoObject</a> .
<a href="#">options</a>	<a href="#">option.Manager</a>	Geo object options manager.  Inherited from <a href="#">GeoObject</a> .
<a href="#">properties</a>	<a href="#">data.Manager</a>	Geo object data manager.  Inherited from <a href="#">GeoObject</a> .

Name	Type	Description
<a href="#">state</a>	<a href="#">data.Manager</a>	<p>State of the geo object. Defined by the following fields:</p> <ul style="list-style-type: none"> <li>• <code>active</code>: Boolean - Indicates that a balloon is open on the geo object.</li> <li>• <code>hover</code>: Boolean - Indicates that the mouse is currently pointed at the geo object.</li> <li>• <code>drag</code>: Boolean - Indicates that the geo object is being dragged</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>

## Events

Name	Description
<a href="#">balloonclose</a>	<p>Closing the balloon. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">balloonopen</a>	<p>Opening a balloon on a geo object. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">beforedrag</a>	<p>Event preceding the "drag" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>position</code> - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• <code>pixelOffset</code> - Array of two numbers that describe the pixel offset at this step.</li> <li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>• <code>setPixelOffset</code> - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent drag event will be canceled.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">beforedragstart</a>	<p>Event preceding the "dragstart" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>position</code> - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">drag</a>	<p>Dragging a geo object. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>position</code> - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• <code>pixelOffset</code> - Array of two numbers that describe the pixel offset at this step.</li> <li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">dragend</a>	<p>End of geo object dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>position</code> - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">dragstart</a>	<p>Start of geo object dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>position</code> - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">editorstatechange</a>	<p>Change in the state of the editor for the geo object's geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code> - Original event of the geometry editor.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the geometry.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">hintclose</a>	<p>Closing the hint. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">hintopen</a>	<p>Opening a hint on a geo object. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldMap</code> - Old map.</li> <li>• <code>newMap</code> - New map.</li> </ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>

Name	Description
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	<p>Returns reference to the map.</p> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	<p>Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	<p>The method provides synchronous access to the overlay.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	<p>Returns link to the parent object, or null if the parent element was not set.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	<p>Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>

## Fields details

### editor

```
{geometryEditor.LineString} editor
```

The "Polyline" geometry editor.

### geometry

```
{geometry.LineString} geometry
```

"Polyline" type of geometry.

## Popup

Extends [IPopup](#).

A class for creating an info object.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
Popup(map[, options])
```

Info object.

### Parameters:

Parameter	Default value	Description
<a href="#">map</a> *	—	Type: <a href="#">Map</a> Reference to the map.
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.closeTimeout</a>	700	Type: Number Delay before closing (in ms).
<a href="#">options.interactivityModel</a>	—	Type: String Key for the interactivity model. Available keys and their values are listed in the description of <a href="#">interactivityModel.storage</a> .
<a href="#">options.openTimeout</a>	150	Type: Number Delay before opening (in ms).
<a href="#">options.pane</a>	—	Type: <a href="#">IPane</a>  String Info object pane. For the list of available default panes, see <a href="#">map.pane.Manager</a> .

Parameter	Default value	Description
<a href="#">options.projection</a>	—	Type: <a href="#">IProjection</a>  The projection of the coordinates to global pixels.
<a href="#">options.zIndex</a>	—	Type: String  The z-index of the info object.

\* Mandatory parameter/option.

### Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .

### Events

Name	Description
<a href="#">close</a>	Closing the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">open</a>	Opening the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .

### Methods

Name	Returns	Description
<a href="#">close([force])</a>	<a href="#">vow.Promise</a>	Closes the info object. Inherited from <a href="#">IPopup</a> .
<a href="#">getData()</a>		Returns info object data. Inherited from <a href="#">IPopup</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object to return the overlay. Inherited from <a href="#">IPopup</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>	Returns the overlay, if one exists. Inherited from <a href="#">IPopup</a> .
<a href="#">getPosition()</a>		Returns the coordinates of the info object. Inherited from <a href="#">IPopup</a> .

Name	Returns	Description
<code>isOpen()</code>	Boolean	Returns the info object state: open/closed. Inherited from <a href="#">IPopup</a> .
<code>open([position[, data]])</code>	<a href="#">vow.Promise</a>	Opens the info object at the specified position. If the info object is already open, it moves it to the specified point. The format and content of the coordinates is determined by the <a href="#">IProjection</a> that is in the options. Inherited from <a href="#">IPopup</a> .
<code>setData(data)</code>	<a href="#">vow.Promise</a>	Defines new data for the info object. Inherited from <a href="#">IPopup</a> .
<code>setPosition(position)</code>	<a href="#">vow.Promise</a>	Specifies a new position for the info object. Inherited from <a href="#">IPopup</a> .

## projection

### projection.Cartesian

Extends [IProjection](#).

Cartesian projection of a rectangular area. Uses the "coordorder" parameter for loading the API - when the value is "latlong", in an array of point coordinates y should be in the first position, and x in the second.

[Constructor](#) | [Methods](#)

#### Constructor

```
projection.Cartesian(bounds[, cycled[, scale]])
```

Creates a projection of a rectangular coordinate area in pixels. The size of the area in pixels is always NxN, where  $N = 256 * 2^{\text{zoom}}$ .

#### Parameters:

Parameter	Default value	Description
<code>bounds</code> *	—	Type: <code>Number[][]</code>  Array of two points, the coordinates of the lower-left and upper-right corners of a rectangular coordinate area.
<code>cycled</code>	<code>[false, false]</code>	Type: <code>Boolean[]</code>  Array of indicators of map cycling on x and y.

Parameter	Default value	Description
<a href="#">scale</a>	1	Type: Number Number[]  The increment of a division on an axis. Can be a number or pair of numbers for each of the axes.

\* Mandatory parameter/option.

## Methods

Name	Returns	Description
<a href="#">fromGlobalPixels(globalPixelPoint, zoom)</a>	Number[]	Converts pixel coordinates to the projection's coordinates at the specified zoom level.  Inherited from <a href="#">IProjection</a> .
<a href="#">getCoordSystem()</a>	<a href="#">ICoordSystem</a>	Returns the coordinate system that is used by the projection.  Inherited from <a href="#">IProjection</a> .
<a href="#">isCycled()</a>	Boolean[]	Indicator of projection cycling.  Inherited from <a href="#">IProjection</a> .
<a href="#">toGlobalPixels(coordPoint, zoom)</a>	Number[]	Converts projection coordinates to global pixel coordinates at the specified zoom level.  Inherited from <a href="#">IProjection</a> .

## projection.sphericalMercator

Static object.

Instance of [IProjection](#)

Mercator projection on a sphere. It is used by many cartographic services, including OpenStreetMap.

### Methods

#### Example:

```
// Creating the map in the spherical Mercator projection
var map = new ymaps.Map('YMapsID', {
  center: [55, 37],
  zoom: 6
}, {
  projection: ymaps.projection.sphericalMercator
});
map.layers.add(new ymaps.Layer('http://tile.openstreetmap.org/%z/%x/%y.png'));
```

## Methods

Name	Returns	Description
<a href="#">fromGlobalPixels(globalPixelPoint, zoom)</a>	Number[]	Converts pixel coordinates to the projection's coordinates at the specified zoom level.

Name	Returns	Description
<a href="#">getCoordSystem()</a>	<a href="#">ICoordSystem</a>	Returns the coordinate system that is used by the projection.
<a href="#">isCycled()</a>	Boolean[]	Indicator of projection cycling.
<a href="#">toGlobalPixels(coordPoint, zoom)</a>	Number[]	Converts projection coordinates to global pixel coordinates at the specified zoom level.

## projection.wgs84Mercator

Static object.

Instance of [IProjection](#)

Mercator projection on a WGS84 reference ellipsoid. Used by Yandex.Maps by default.

### Methods

#### Methods

Name	Returns	Description
<a href="#">fromGlobalPixels(globalPixelPoint, zoom)</a>	Number[]	Converts pixel coordinates to the projection's coordinates at the specified zoom level.
<a href="#">getCoordSystem()</a>	<a href="#">ICoordSystem</a>	Returns the coordinate system that is used by the projection.
<a href="#">isCycled()</a>	Boolean[]	Indicator of projection cycling.
<a href="#">toGlobalPixels(coordPoint, zoom)</a>	Number[]	Converts projection coordinates to global pixel coordinates at the specified zoom level.

## ready

Static function.

Performs the passed function when the API and DOM are ready for use.

**Returns** the promise object that is verified by the API namespace, or rejected if an error occurred when loading.

```
{ vow.Promise } ready([successCallback[, errorCallback[, context]])
```

### Parameters:

Parameter	Default value	Description
<a href="#">successCallback</a>	—	<p>Type: Function Object</p> <p>Function that will be called after successfully loading and initializing the API and DOM, or an object with parameters if the extended syntax is used.</p> <p>Available parameters:</p> <ul style="list-style-type: none"> <li>require — Array of additional modules that must be loaded with the API.</li> <li>successCallback - Function that will be called when the API is loaded successfully.</li> <li>errorCallback - Function that will be called if an error occurs.</li> <li>context - Execution context for functions.</li> </ul> <p>All parameters are optional.</p> <p>The API namespace will be passed to successCallback.</p>
<a href="#">errorCallback</a>	—	<p>Type: Function</p> <p>The function that will be called if an error occurred during initialization. The error will be passed to the function.</p>
<a href="#">context</a>	—	<p>Type: Object</p> <p>Context for the function.</p>

## Examples:

### 1.

```
<!DOCTYPE html>
<html>
<head>
  <title>Example</title>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <script src="http://api-maps.yandex.ru/2.1/?apikey=<your API key>&lang=ru_RU" type="text/javascript"></script>
  <script type="text/javascript">
    ymaps.ready(function () {
      var map = new ymaps.Map('map', {
        center: [55.7, 37.6],
        zoom: 10
      });
      // ...
    });
  </script>
</head>
<body>
  <div id="map" style="width: 500px; height: 500px;"></div>
</body>
</html>
```

### 2.

```
// Example using the extended syntax.
ymaps.ready({
  // successCallback will be called when the API and the "myModule1" module are loaded.
  require: ['myModule1'],
  successCallback: function (ym) {
    var map = new ymaps.Map('map', {
      center: [55.7, 37.6],
      zoom: 10
    });
  });
```

```

var obj = new ymaps.myModule1();
// ...
}
})

```

## Rectangle

Extends [GeoObject](#).

Rectangle. A geo object with the geometry [geometry.Rectangle](#).

See [GeoObject](#) [geometry.Rectangle](#)

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
Rectangle(geometry[, properties[, options]])
```

Creates a rectangle instance.

#### Parameters:

Parameter	Default value	Description
<a href="#">geometry</a> *	—	<p>Type: <code>Number[][]</code>            Object <a href="#">IRectangleGeometry</a></p> <p>Coordinates of two opposite corners, a hash object with geometry parameters, or a reference to the rectangle geometry object.</p>
<a href="#">properties</a>	—	<p>Type: <code>Object</code> <a href="#">IDataManager</a></p> <p>Rectangle data. Can be set as a class instance implementing the <a href="#">IDataManager</a> interface, or as a hash. When options are set to default values, the following data fields are interpreted by a geo object:</p> <ul style="list-style-type: none"> <li><code>hintContent</code> - Content of the rectangle's popup hint.</li> <li><code>balloonContent</code> - Content of the rectangle's balloon.</li> <li><code>balloonContentHeader</code> - Content of the rectangle balloon title.</li> <li><code>balloonContentBody</code> - Content of the main part of the rectangle's balloon.</li> <li><code>balloonContentFooter</code> - Content of the lower part of the rectangle's balloon.</li> </ul> <p>The <code>balloonContent</code> field is a shortcut for the <code>balloonContentBody</code> field, but if they are both set simultaneously, <code>balloonContentBody</code> takes priority. You can also add your own custom fields to the rectangle data and use them, for example, in the balloon layout.</p>

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object  Rectangle options. Using this parameter, you can set options for the rectangle itself, as well as for its parts: <ul style="list-style-type: none"> <li>Options for the rectangle's <a href="#">balloon</a> with the <code>balloon</code> prefix.</li> <li>Options for the rectangle's <a href="#">popup</a> hint with the <code>hint</code> prefix.</li> <li>Geometry options can be set without a prefix. See the description of the <a href="#">IGeometry</a> class for the <a href="#">geometry.Rectangle</a> geometry.</li> </ul>
<a href="#">options.cursor</a>	"pointer"	Type: String  Type of cursor over a rectangle.
<a href="#">options.draggable</a>	false	Type: Boolean  Checks whether the rectangle can be dragged.
<a href="#">options.fill</a>	true	Type: Boolean  Whether the shape is filled.
<a href="#">options.fillColor</a>	"0066ff99"	Type: String  Fill color.
<a href="#">options.fillImageHref</a>	—	Type: String  Background image. When this option is enabled in <b>stretch</b> mode, the "fillColor" value is ignored.
<a href="#">options.fillMethod</a>	'stretch'	Type: String  Type of background fill. Accepts one of two values: <ul style="list-style-type: none"> <li><code>stretch</code> - The background image stretches to fit the size of the overlay.</li> <li><code>tile</code> - The background image is repeated, without changing its size. This is similar to <code>background-repeat</code> in CSS. It can be used for filling a shape with a template.</li> </ul>
<a href="#">options.fillOpacity</a>	1	Type: Number  Fill transparency.
<a href="#">options.hasBalloon</a>	true	Type: Boolean  Checks if the rectangle has the "balloon" field.

Parameter	Default value	Description
<a href="#">options.hasHint</a>	true	Type: Boolean  Checks if the rectangle has the "hint" field.
<a href="#">options.interactiveZIndex</a>	false	Type: Boolean  Enables automatically modifying the z-index of the rectangle depending on its state.
<a href="#">options.interactivityModel</a>	"default#geoObject"	Type: String  Interactivity model. Available keys and their values are listed in the description of <a href="#">interactivityModel.storage</a> .
<a href="#">options.opacity</a>	1	Type: Number  Transparency.
<a href="#">options.openBalloonOnClick</a>	true	Type: Boolean  Checks whether to show the balloon when the rectangle is clicked on.
<a href="#">options.openEmptyBalloon</a>	false	Type: Boolean  Checks whether to show an empty balloon when the rectangle is clicked on.
<a href="#">options.openEmptyHint</a>	false	Type: Boolean  Checks whether to show an empty hint when the mouse pointer hovers over the rectangle.
<a href="#">options.openHintOnHover</a>	true	Type: Boolean  Checks whether to show an empty hint when the mouse pointer hovers over the rectangle.
<a href="#">options.outline</a>	true	Type: Boolean  Whether the shape has an outline.
<a href="#">options.pane</a>	"places"	Type: String  The key of the pane where the rectangle overlay is placed.

Parameter	Default value	Description
<a href="#">options.rectangleOverlay</a>	"default#rectangle"	<p>Type: <code>String Function</code></p> <p>Key identifier from <a href="#">overlay.storage</a> or the overlay class. The generator function accepts three parameters:</p> <ul style="list-style-type: none"> <li>geometry: <a href="#">IPixelCircleGeometry</a> - The pixel geometry itself.</li> <li>data: <a href="#">IDataManager</a> or Object - Overlay data.</li> <li>options: Object - The overlay options.</li> </ul> <p>And returns <a href="#">vow.Promise</a>.</p>
<a href="#">options.strokeColor</a>	"0066ffff"	<p>Type: <code>String String[]</code></p> <p>Color of the line or outline. You can set multiple values for a multistroke outline.</p>
<a href="#">options.strokeOpacity</a>	1	<p>Type: <code>Number Number[]</code></p> <p>Transparency of the line or outline. You can set multiple values for a multistroke outline.</p>
<a href="#">options.strokeStyle</a>	—	<p>Type: <code>String Object String[] Object[]</code></p> <p>Style of the line or outline. You can set multiple values for a multistroke outline.</p>
<a href="#">options.strokeWidth</a>	1	<p>Type: <code>Number Number[]</code></p> <p>Thickness of the line or outline. You can set multiple values for a multistroke outline.</p>
<a href="#">options.syncOverlayInit</a>	false	<p>Type: <code>Boolean</code></p> <p>Enables synchronously adding an overlay to the map. By default, overlays are added to the map asynchronously to prevent the browser from hanging when adding a large number of geo objects. However, adding asynchronously does not allow accessing the overlay immediately after adding a rectangle to the map.</p>
<a href="#">options.useMapMarginInDragging</a>	true	<p>Type: <code>Boolean</code></p> <p>When an object is dragged to the edge of the map, the map center changes automatically. Whether to use map margins when automatically shifting the map center with <a href="#">map.margin.Manager</a>.</p>
<a href="#">options.visible</a>	true	<p>Type: <code>Boolean</code></p> <p>Checks the visibility of the rectangle.</p>

Parameter	Default value	Description
<a href="#">options.zIndex</a>	—	Type: Number  The z-index of the rectangle in the normal state. Lowest priority.
<a href="#">options.zIndexActive</a>	—	Type: Number  The z-index of the rectangle with an opened balloon. Highest priority.
<a href="#">options.zIndexDrag</a>	—	Type: Number  The z-index of the rectangle while dragging.
<a href="#">options.zIndexHover</a>	—	Type: Number  The z-index of a rectangle when the mouse pointer is hovering over it.

\* Mandatory parameter/option.

#### Example:

```
// Creating a geodesic circle with a radius of 1000 kilometers.
var circle = new ymaps.Circle([[50, 50], 1000000], {}, {
  draggable: true
});
// Adding the circle to the map.
myMap.geoObjects.add(circle);

// Creating a rectangle based on the circle's boundaries.
var rectangle = new ymaps.Rectangle(circle.geometry.getBounds(), {}, {
  fill: false,
  coordRendering: "boundsPath",
  strokeWidth: 4
});
// Adding the rectangle to the map.
myMap.geoObjects.add(rectangle);

// Updating the rectangle's coordinates when changing the circle geometry.
circle.geometry.events.add("change", function (event) {
  this.geometry.setCoordinates(event.get("target").getBounds());
}, rectangle);
```

#### Fields

Name	Type	Description
<a href="#">balloon</a>	<a href="#">geoObject.Balloon</a>	Balloon for a geo object.  Inherited from <a href="#">GeoObject</a> .
<a href="#">editor</a>	null	An editor for the "Rectangle" geometry has not yet been implemented.
<a href="#">events</a>	<a href="#">event.Manager</a>	Event manager.  Inherited from <a href="#">GeoObject</a> .
<a href="#">geometry</a>	<a href="#">geometry.Rectangle</a>	The "Rectangle" type of geometry.
<a href="#">hint</a>	<a href="#">geoObject.Hint</a>	Geo object hint.  Inherited from <a href="#">GeoObject</a> .
<a href="#">options</a>	<a href="#">option.Manager</a>	Geo object options manager.  Inherited from <a href="#">GeoObject</a> .

Name	Type	Description
<a href="#">properties</a>	<a href="#">data.Manager</a>	Geo object data manager. Inherited from <a href="#">GeoObject</a> .
<a href="#">state</a>	<a href="#">data.Manager</a>	State of the geo object. Defined by the following fields: <ul style="list-style-type: none"> <li>• active: Boolean - Indicates that a balloon is open on the geo object.</li> <li>• hover: Boolean - Indicates that the mouse is currently pointed at the geo object.</li> <li>• drag: Boolean - Indicates that the geo object is being dragged</li> </ul> Inherited from <a href="#">GeoObject</a> .

## Events

Name	Description
<a href="#">balloonclose</a>	Closing the balloon. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">GeoObject</a> .
<a href="#">balloonopen</a>	Opening a balloon on a geo object. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">GeoObject</a> .
<a href="#">beforedrag</a>	Event preceding the "drag" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>• position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• pixelOffset - Array of two numbers that describe the pixel offset at this step.</li> <li>• domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> Names of methods that are accessible via <a href="#">Event.callMethod</a> : <ul style="list-style-type: none"> <li>• setPixelOffset - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.</li> </ul> If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent drag event will be canceled. Inherited from <a href="#">GeoObject</a> .
<a href="#">beforedragstart</a>	Event preceding the "dragstart" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>• position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> If the <a href="#">Event.preventDefault</a> method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled. Inherited from <a href="#">GeoObject</a> .
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">drag</a>	<p>Dragging a geo object. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>pixelOffset - Array of two numbers that describe the pixel offset at this step.</li> <li>domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">dragend</a>	<p>End of geo object dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">dragstart</a>	<p>Start of geo object dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">editorstatechange</a>	<p>Change in the state of the editor for the geo object's geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>originalEvent - Original event of the geometry editor.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">hintclose</a>	<p>Closing the hint. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">hintopen</a>	<p>Opening a hint on a geo object. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"> <li>oldMap - Old map.</li> <li>newMap - New map.</li> </ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>

Name	Description
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	<p>Returns reference to the map.</p> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	<p>Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	<p>The method provides synchronous access to the overlay.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	<p>Returns link to the parent object, or null if the parent element was not set.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	<p>Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>

## Fields details

### editor

```
{null} editor
```

An editor for the "Rectangle" geometry has not yet been implemented.

### geometry

```
{geometry.Rectangle} geometry
```

The "Rectangle" type of geometry.

## regions

### regions.load



**Attention:** This function is deprecated. Use [borders.load](#).

Static function.

Provides access to the geometry of various regions and countries.

**Returns** Promise object.

```
{ vow.Promise } regions.load(region[, options])
```

#### Parameters:

Parameter	Default value	Description
<a href="#">region</a> *	—	Type: String  The ISO_3166-1 country code (RU, UA, BY, KZ) for loading the regional area, or '001' for loading the geometry of country borders.
<a href="#">options</a>	—	Type: Object  Display options.
<a href="#">options.disputedBorders</a>	—	Type: String  Two-letter code of the country to use as the official reference for determining the administrative subordination of disputed territories. Accepted values: 'RU', 'UA', 'BY', 'KZ'. By default, it coincides with the country code that is specified when loading the API. Unsupported country codes are reset to RU. For the region '001' (borders of countries), the code 'UN' is supported — world borders according to the United Nations.
<a href="#">options.lang</a>	—	Type: String  Language (ru, uk, en, be).

Parameter	Default value	Description
<code>options.quality</code>	1	Type: Number  Quality level. Available values: <ul style="list-style-type: none"><li>• 0 - minimal quality</li><li>• 1 - standard quality</li><li>• 2 - improved quality</li><li>• 3 - high quality</li></ul> The quality level affects how accurately curves are represented, as well as the volume of the data file.

\* Mandatory parameter/option.

#### Example:

```
ymaps.regions.load('RU', {
  lang: 'en'
}).then(function (result) {
  geoMap.geoObjects.add(result.geoObjects);
});
```

## RemoteObjectManager

Extends [ICustomizable](#), [IEventEmitter](#), [IGeoObject](#), [IParentOnMap](#).

The object manager that optimizes downloading of objects from the server. The manager sends data requests to the specified url in the JSONP format. This format corresponds to the format of objects added to [ObjectManager](#), [ObjectManager.add](#). Objects of type "Cluster" with the following fields are also supported:

- type - The object type, always "Cluster" for clusters.
- id - Unique cluster ID.
- geometry - The cluster geometry in JSON format.
- features - An array of cluster objects. Optional field.
- bbox - An array of coordinates describing a rectangular area that includes all the cluster objects.
- number - Number of objects in the cluster.
- properties - Cluster data.

This module is designed for downloading and displaying the data which was previously processed on the server. Particularly, it is recommended to use the module for displaying results of server-side clustering. Data is requested again if the map zoom level is changed. The module does not provide clustering or filtering possibilities for managing visibility on the client side. To cluster objects on the client side after loading, use [LoadingObjectManager](#). Note that objects drawn on the map via this manager can't have editing and dragging modes enabled.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
RemoteObjectManager(urlTemplate[, options])
```

### Parameters:

Parameter	Default value	Description
<a href="#">uriTemplate</a> *	—	Type: String  URL data template. Supports special constructions similar to <a href="#">Layer</a> . Substitutions are also supported: <ul style="list-style-type: none"><li>• %b is replaced by an array of geographic coordinates that describes the rectangular region for which you want to load data.</li><li>• %t is replaced by an array of tile numbers that describes the rectangular region to load data for.</li></ul>
<a href="#">options</a>	—	Type: Object  Options. <ul style="list-style-type: none"><li>• You can set all the options specified in the <a href="#">Clusterer</a> description, except for hasBalloon and hasHint options.</li><li>• Cluster options are set with the "cluster" prefix. The list of options is specified in the description of <a href="#">ClusterPlacemark</a>;</li><li>• Options for singular objects should be specified with the geoObject prefix. The list of options is specified in <a href="#">GeoObject</a>. Note that the manager ignores the 'visible' option.</li></ul>
<a href="#">options.loadTileSize</a>	256	Type: Number  Tile size for data loading.
<a href="#">options.paddingParamName</a>	'callback'	Type: Boolean  Name of the GET parameter that contains the value of the JSONP callback.

Parameter	Default value	Description
<a href="#">options.paddingTemplate</a>	null	<p>Type: String</p> <p>Template for a jsonp callback. Supports the same substitutions as uriTemplate. All characters other than letters and numbers will be replaced with '_'. If the parameter is omitted, the name of the jsonp callback will be generated automatically. Conversion examples for tileNumber=[3, 1], zoom=9:</p> <ul style="list-style-type: none"> <li>'myCallback=%x' =&gt; 'myCallback_3'</li> <li>'%c' =&gt; 'x_3_y_1_z_9'</li> <li>'callback2_%c' =&gt; 'callback2_x_3_y_1_z_9'</li> <li>'callback%test' =&gt; 'callback_test'</li> <li>'callback_%b' =&gt; 'callback_85_0841__180_0000_85_0841_180_0000'</li> </ul> <p>Note that if substitution options are not used in the value, this may lead to an error. All requests will go to the same callback function.</p>
<a href="#">options.splitRequests</a>	false	<p>Type: Boolean</p> <p>Divide requests for data into requests for individual tiles. By default, requests are made for data for a rectangular region that contains multiple tiles.</p>
<a href="#">options.syncOverlayInit</a>	false	<p>Type: Boolean</p> <p>A flag that allows creating overlays for objects synchronously. Note that when you create an overlay synchronously, you should ensure that the appropriate class, which implements the IOverlay interface, is loaded. By default, the overlays are created asynchronously, and the overlay class is loaded on demand.</p>

\* Mandatory parameter/option.

## Examples:

### 1.

```
var objectManager = new ymaps.RemoteObjectManager('http://myServer.com/tile?bbox=%b', {
  // Cluster options are set with the "cluster" prefix.
  clusterHasBalloon: false,
  // Geo object options are set with the "geoObject" prefix.
  geoObjectOpenBalloonOnClick: false
});

// You can set options directly for child collections.
objectManager.clusters.set({
  preset: 'islands#grayClusterIcons',
  hintContentLayout: ymaps.templateLayoutFactory.createClass('Group of objects')
});
objectManager.objects.set('preset', 'islands#grayIcon');
```

### 2.

```
An example of RemoteObjectManager response
jsonp_callback({
  // The response contains the error and data fields. If an error occurs, the "error" field
```

```

// contains the error code or description.
error: null,
data: {
  type: 'FeatureCollection',
  features: [
    {
      type: 'Feature',
      geometry: {
        type: 'Point',
        coordinates: [55, 35]
      },
      id: 23,
      properties: {
        balloonContent: 'Placemark balloon content',
        iconContent: 'Placemark content'
      },
      options: {
        preset: 'islands#yellowIcon'
      }
    },
    {
      type: 'Cluster',
      id: 24,
      bbox: [[35, 46], [46, 57]],
      number: 34,
      // Array describing the 34 objects in the cluster.
      // Optional field.
      // If omitted, an empty balloon opens when the cluster is clicked.
      features: [{
        type: 'Feature',
        id: 512,
        properties: {
          balloonContent: 'Placemark balloon content',
          clusterCaption: 'Placemark title in the cluster balloon'
        },
        ...
      },
      ],
      geometry: {
        type: 'Point',
        coordinates: [40.5, 51]
      },
      properties: {
        iconContent: 34
      }
    }
  ]
}
});

```

## Fields

Name	Type	Description
<a href="#">clusters</a>	<a href="#">objectManager.ClusterCollection</a>	Collection of clusters generated by the manager.
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">objects</a>	<a href="#">objectManager.ObjectCollection</a>	Collection of objects added to the layer.
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">IDataManager</a>	Geo object data. Inherited from <a href="#">IGeoObject</a> .
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

## Events

Name	Description
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dataloaderror</a>	<p>An error occurred when loading data. Instance of the <a href="#">Event</a> class.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li></ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>• oldMap - Old map.</li><li>• newMap - New map.</li></ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getBounds()</a>	Number[][] null	Calculates the boundaries in geo coordinates for an area that covers all the loaded objects in the manager.
<a href="#">getMap()</a>	Map	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getObjectState(id)</a>	Object	Getting information about the current state of an object added to the manager.
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">getPixelBounds()</a>	Number[][] null	Calculates the boundaries in global pixel coordinates for an area that covers all the loaded objects in the manager.
<a href="#">getTileUrl()</a>	String null	Returns URL of the tile with data.
<a href="#">getUrlTemplate()</a>	String	Returns URL data template.
<a href="#">reloadData()</a>		Method that deletes all previously loaded data and sends a request for new data.
<a href="#">setFilter(filterFunction)</a>		Sets a filter function for objects. Filters both individual objects and clusters.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">setUrlTemplate(urlTemplate)</a>		

## Fields details

### clusters

```
{objectManager.ClusterCollection} clusters
```

Collection of clusters generated by the manager.

#### Example:

```
objectManager.clusters.events.add('click', function (e) {  
  var objectId = e.get('objectId');  
  objectManager.clusters.balloon.open(objectId);  
});
```

### objects

```
{objectManager.ObjectCollection} objects
```

Collection of objects added to the layer.

#### Example:

```
objectManager.objects.events.add('click', function (e) {  
  var objectId = e.get('objectId');  
  objectManager.objects.balloon.open(objectId);  
});
```

## Events details

### dataloadererror

An error occurred when loading data. Instance of the [Event](#) class.

## Methods details

### getBounds

```
{Number[][]|null} getBounds()
```

Calculates the boundaries in geo coordinates for an area that covers all the loaded objects in the manager.

**Returns** array of the area's coordinates, or null if the manager has not been added to the map.

### getObjectState

```
{Object} getObjectState(id)
```

Getting information about the current state of an object added to the manager.

**Returns** object with following fields:

- **found** - Attribute that indicates whether an object with the passed ID exists in the loaded data. Type: Boolean.
- **isShown** - Attribute that indicates whether the object is located in the visible area of the map. Type: Boolean.
- **isFilteredOut** - Attribute indicating whether an object passed through filtration. If the filter is not set or the object passed through filtration, the value of the field is "false". Type: Boolean.

**Parameters:**

Parameter	Default value	Description
<code>id</code> *	—	Type: Object  ID of the object to get the state for.

\* Mandatory parameter/option.

#### Example:

```
remoteObjectManager.setFilter('properties.type == "shop"');
...
if (!remoteObjectManager.getObjectState(7).isFilteredOut) {
    remoteObjectManager.objects.balloon.open(7);
}
```

### getPixelBounds

```
{Number[][]|null} getPixelBounds()
```

Calculates the boundaries in global pixel coordinates for an area that covers all the loaded objects in the manager.

**Returns** array of the area's coordinates, or null if the manager has not been added to the map.

### getTileUrl

```
{String|null} getTileUrl()
```

**Returns** URL of the tile with data.

#### Parameters:

Parameter	Default value	Description
<code>parameters</code> *	—	Type:

\* Mandatory parameter/option.

#### Example:

```
var objectManager = new ymaps.RemoteObjectManager('http://myServer.com/tile?bbox=%b');
objectManager.getTileUrl = function (parameters) {
    var boundingBox = parameters.boundingBox.join('~');
    return this.getUrlTemplate().replace(/%b/g, boundingBox);
};
```

### getUrlTemplate

```
{String} getUrlTemplate()
```

**Returns** URL data template.

### reloadData

```
{} reloadData()
```

Method that deletes all previously loaded data and sends a request for new data.

### setFilter

```
{} setFilter(filterFunction)
```

Sets a filter function for objects. Filters both individual objects and clusters.

#### Parameters:

Parameter	Default value	Description
<code>filterFunction</code> *	—	<p>Type: Function String</p> <p>Filter function. Takes a single object added to <code>ObjectManager</code>. If the function returns true, the object will be processed. If false, the object will be excluded from further processing. A string can also be passed as a filter. The following keywords are available in the string filter:</p> <ul style="list-style-type: none"> <li><code>options</code> - Accessing the object's options.</li> <li><code>properties</code> - Accessing the object's data.</li> <li><code>geometry</code> - Accessing the object's geometry.</li> <li><code>id</code> - Accessing the object's identifier.</li> </ul> <p>For a filter, you can specify an expression that returns true or false.</p>

\* Mandatory parameter/option.

#### Examples:

##### 1.

```
// Select clusters with id > 100.
objectManager.setFilter('object.type == "Cluster" && id > 100');
```

##### 2.

```
// Only objects of the specified types will be displayed on the map.
objectManager.setFilter('properties.type == "cafe" || properties.type == "pharmacy");
```

##### 3.

```
// You can define a filter function.
objectManager.setFilter(function (object) {
    return object.properties.name != 'The one who cannot be shown.';
});
```

#### setUrlTemplate

```
{ } setUrlTemplate(urlTemplate)
```

#### Parameters:

Parameter	Default value	Description
<code>urlTemplate</code> *	—	<p>Type: String</p> <p>URL data template.</p>

\* Mandatory parameter/option.

## route

Static function.

Plots a route through the specified points.

**Returns** a promise object that is confirmed by the route object [router.Route](#), or the multi-stop route object [multiRouter.MultiRoute](#), depending on the value of the multiRoute parameter. If an error occurs, the promise object is rejected.

```
{ vow.Promise } route(points[, params])
```

### Parameters:

Parameter	Default value	Description
<a href="#">points</a> *	—	Type: Object[]  Array of points that the route should go through. Each point can be set by a string containing the address, an array of the coordinates, and a JSON object with the following fields: <ul style="list-style-type: none"> <li>type: String - Type of point. The 'wayPoint' value sets the route waypoint. Use the "viaPoint" value to define a throughpoint, i.e. a point that must be driven through without stopping.</li> <li>point: Number[] String - Array of the coordinates of a point, or its address as a string.</li> </ul>
<a href="#">params</a>	—	Type: Object  Routing options.
<a href="#">params.avoidTrafficJams</a>	false	Type: Boolean  Enables constructing a route with consideration for traffic. When using the options, keep in mind that it is not always possible to detour around traffic jams.
<a href="#">params.boundedBy</a>	—	Type: Number[][]  Area on the map where the objects being searched for are presumably located. This is used if the route points are set using the mailing address instead of coordinates.
<a href="#">params.mapStateAutoApply</a>	false	Type: Boolean  Flag that allows to automatically set the center and map zoom so that the route will be entirely visible.

Parameter	Default value	Description
<a href="#">params.multiRoute</a>	false	Type: Boolean  Allows to construct multi-stop routes.
<a href="#">params.reverseGeocoding</a>	false	Type: Boolean  Whether to use reverse geocoding for points specified as coordinates.
<a href="#">params.routingMode</a>	"auto"	Type: String  Routing type. Accepts one of three string values: <ul style="list-style-type: none"> <li>"auto" - Driving route.</li> <li>"masstransit" - Routing using public transit. Only for multi-stop routes (the multiRoute option must be set to true).</li> <li>"pedestrian" - Walking route. Only for multi-stop routes (the multiRoute option must be set to true).</li> </ul>
<a href="#">params.searchCoordOrder</a>	—	Type: String  Defines how to interpret the coordinates in the request. This is used if the route points are set using coordinates, and not the mailing address.
<a href="#">params.strictBounds</a>	false	Type: Boolean  Search only inside the area defined by the "boundedBy" option.
<a href="#">params.useMapMargin</a>	true	Type: Boolean  Whether to account for map margins <a href="#">map.margin.Manager</a> .
<a href="#">params.viaIndexes</a>	[]	Type: Integer[]  Indexes of the multi-stop route throughpoints.
<a href="#">params.zoomMargin</a>	0	Type: Number Number[]  Offset from the map viewport borders when changing the zoom level. If a single number is set, it is applied to each side. If two numbers are set, they are the horizontal and vertical margins, respectively. If an array of four numbers is set, they are the top, right, bottom, and left margins. When the "useMapMargin" parameter is enabled, the "zoomMargin" value is combined with the values that were calculated in the margins manager <a href="#">map.margin.Manager</a> .

\* Mandatory parameter/option.

## Examples:

### 1.

```
// Building the route from Korolev to Krasnogorsk via Khimki and Mytischki,
// where Mytischki is a throughpoint. Setting coordinates for Krasnogorsk.
ymaps.route([
  'Korolev',
  { type: 'viaPoint', point: 'Mytischki' },
  'Himki',
  { type: 'wayPoint', point: [55.811511, 37.312518] }
], {
  mapStateAutoApply: true
}).then(function (route) {
  route.getPaths().options.set({
    // the balloon only shows information about the travel time with traffic
    balloonContentLayout: ymaps.templateLayoutFactory.createClass('{{ properties.humanJamsTime }}'),
    // you can make settings for route graphics
    strokeColor: '0000ffff',
    opacity: 0.9
  });
  // adding the route to the map
  map.geoObjects.add(route);
});
```

### 2.

```
// Building a multi-stop route and adding it to the map using autoscaling.
ymaps.route(['Southern Butovo', 'Moscow, metro Park Kultury'], {
  multiRoute: true
}).done(function (route) {
  route.options.set("mapStateAutoApply", true);
  myMap.geoObjects.add(route);
}, function (err) {
  throw err;
}, this);
```

## router

### router.addon

#### router.addon.editor router.addon.editor.get

Static function.

Returns router editor.

```
{ router.Route } router.addon.editor.get()
```

#### Example:

```
ymaps.router.addon.editor.get(myMap)
```

### router.Editor

**Note:** The constructor of the `router.Editor` class is hidden, as this class is not intended for autonomous initialization.

Extends [ICustomizable](#), [IEventEmitter](#).

Route editor. The constructor is not available in the `package.full` (a standard set of modules). This module is loaded on demand.

[Fields](#) | [Events](#) | [Methods](#)

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">state</a>	<a href="#">IDataManager</a>	The state manager of the route editor. Data fields that are available via the "get" and "set" methods: <ul style="list-style-type: none"> <li><code>routeloading</code>: Boolean - Flag for whether the data is currently being downloaded from the routing service.</li> <li><code>waypointsdrag</code>: Boolean - Flag for whether the waypoint is currently being dragged.</li> <li><code>viapointsdrag</code>: Boolean - Flag for whether the throughpoint is currently being dragged.</li> </ul>

**Events**

Name	Description
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">routeupdate</a>	Updating the route. Using the value of the <code>e.get('rough')</code> flag, you can determine whether the event was thrown after editing was completed or during it. If you want your application to update information associated with the route, you need to make a check for <code>e.get('rough') == false</code> in order to avoid handling this event too often.
<a href="#">start</a>	Enabling the editor.
<a href="#">stop</a>	Disabling the editor.
<a href="#">viapointadd</a>	Adding a throughpoint. Use <code>e.get('viaPoint')</code> to get a throughpoint to be added.
<a href="#">viapointdragend</a>	End of throughpoint dragging. Use <code>e.get('viaPoint')</code> to get a throughpoint.
<a href="#">viapointdragstart</a>	Start of throughpoint dragging. Use <code>e.get('viaPoint')</code> to get a throughpoint.
<a href="#">viapointremove</a>	Deleting a throughpoint. Use <code>e.get('viaPoint')</code> to get a throughpoint to be deleted.
<a href="#">waypointadd</a>	Adding a waypoint. Use <code>e.get('wayPoint')</code> to get a waypoint to be added.
<a href="#">waypointdragend</a>	End of waypoint dragging. Use <code>e.get('wayPoint')</code> to get a waypoint.
<a href="#">waypointdragstart</a>	Start of waypoint dragging. Use <code>e.get('wayPoint')</code> to get a waypoint.
<a href="#">waypointremove</a>	Deleting a waypoint. Use <code>e.get('wayPoint')</code> to get a waypoint to be deleted.

**Methods**

Name	Description
<a href="#">start([options])</a>	Enables the route editor.
<a href="#">stop()</a>	Disables the route editor.

## Fields details

### state

```
{IDataManager} state
```

The state manager of the route editor.

Data fields that are available via the "get" and "set" methods:

- `routeloading`: Boolean - Flag for whether the data is currently being downloaded from the routing service.
- `waypointsdrag`: Boolean - Flag for whether the waypoint is currently being dragged.
- `viapointsdrag`: Boolean - Flag for whether the throughpoint is currently being dragged.

## Events details

### routeupdate

Updating the route. Using the value of the `e.get('rough')` flag, you can determine whether the event was thrown after editing was completed or during it. If you want your application to update information associated with the route, you need to make a check for `e.get('rough') == false` in order to avoid handling this event too often.

### start

Enabling the editor.

### stop

Disabling the editor.

### viapointadd

Adding a throughpoint. Use `e.get('viaPoint')` to get a throughpoint to be added.

### viapointdragend

End of throughpoint dragging. Use `e.get('viaPoint')` to get a throughpoint.

### viapointdragstart

Start of throughpoint dragging. Use `e.get('viaPoint')` to get a throughpoint.

### viapointremove

Deleting a throughpoint. Use `e.get('viaPoint')` to get a throughpoint to be deleted.

### waypointadd

Adding a waypoint. Use `e.get('wayPoint')` to get a waypoint to be added.

### waypointdragend

End of waypoint dragging. Use `e.get('wayPoint')` to get a waypoint.

### waypointdragstart

Start of waypoint dragging. Use `e.get('wayPoint')` to get a waypoint.

### waypointremove

Deleting a waypoint. Use `e.get('wayPoint')` to get a waypoint to be deleted.

## Methods details

### start

```
{ } start([options])
```

Enables the route editor.

#### Parameters:

Parameter	Default value	Description
<code>options</code>	—	Type: Object Options.
<code>options.addViaPoints</code>	true	Type: Boolean If true, adding throughpoints is allowed; if false, it is prohibited.
<code>options.addWayPoints</code>	false	Type: Boolean If true, waypoints can be added when clicking on the map; if false, this is prohibited.
<code>options.editViaPoints</code>	true	Type: Boolean If true, editing (moving) throughpoints is allowed; if false, it is prohibited.
<code>options.editWayPoints</code>	true	Type: Boolean If true, editing (moving) waypoints is allowed; if false, it is prohibited.
<code>options.removeViaPoints</code>	true	Type: Boolean If true, deleting throughpoints by double-click is allowed; if false, it is prohibited.
<code>options.removeWayPoints</code>	false	Type: Boolean If true, deleting waypoints by double-click is allowed; if false, it is prohibited.

### stop

```
{ } stop()
```

Disables the route editor.

## router.Path

**Note:** The constructor of the `router.Path` class is hidden, as this class is not intended for autonomous initialization.

Extends [GeoObject](#).

Object that describes part of the route (a path). The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand. The route can contain several paths, and each path connects two waypoints.

**See** [route](#)

[Fields](#) | [Events](#) | [Methods](#)

## Fields

Name	Type	Description
<a href="#">balloon</a>	<a href="#">geoObject.Balloon</a>	Balloon for a geo object. Inherited from <a href="#">GeoObject</a> .
<a href="#">editor</a>	<a href="#">IGeometryEditor</a>	Editor for the geo object geometry. Inherited from <a href="#">GeoObject</a> .
<a href="#">events</a>	<a href="#">event.Manager</a>	Event manager. Inherited from <a href="#">GeoObject</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">GeoObject</a> .
<a href="#">hint</a>	<a href="#">geoObject.Hint</a>	Geo object hint. Inherited from <a href="#">GeoObject</a> .
<a href="#">options</a>	<a href="#">option.Manager</a>	Geo object options manager. Inherited from <a href="#">GeoObject</a> .
<a href="#">properties</a>	<a href="#">data.Manager</a>	Geo object data manager. Inherited from <a href="#">GeoObject</a> .
<a href="#">state</a>	<a href="#">data.Manager</a>	State of the geo object. Defined by the following fields: <ul style="list-style-type: none"> <li>• active: Boolean - Indicates that a balloon is open on the geo object.</li> <li>• hover: Boolean - Indicates that the mouse is currently pointed at the geo object.</li> <li>• drag: Boolean - Indicates that the geo object is being dragged</li> </ul> Inherited from <a href="#">GeoObject</a> .

## Events

Name	Description
<a href="#">balloonclose</a>	Closing the balloon. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">GeoObject</a> .
<a href="#">balloonopen</a>	Opening a balloon on a geo object. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">GeoObject</a> .

Name	Description
<a href="#">beforedrag</a>	<p>Event preceding the "drag" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>pixelOffset - Array of two numbers that describe the pixel offset at this step.</li> <li>domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>setPixelOffset - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent drag event will be canceled.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">beforedragstart</a>	<p>Event preceding the "dragstart" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">drag</a>	<p>Dragging a geo object. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>pixelOffset - Array of two numbers that describe the pixel offset at this step.</li> <li>domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">dragend</a>	<p>End of geo object dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>

Name	Description
<a href="#">dragstart</a>	<p>Start of geo object dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>position</code> - Coordinates relative to the document. Array in the format [pageX, pageY].</li><li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li></ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">editorstatechange</a>	<p>Change in the state of the editor for the geo object's geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>originalEvent</code> - Original event of the geometry editor.</li></ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the geometry.</li></ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">hintclose</a>	<p>Closing the hint. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">hintopen</a>	<p>Opening a hint on a geo object. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>• <code>oldMap</code> - Old map.</li><li>• <code>newMap</code> - New map.</li></ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getHumanJamsTime()</a>	String	Returns a string representation of path driving time with measurement units, with consideration for traffic.
<a href="#">getHumanLength()</a>	String	Returns a string representation of path length with measurement units.
<a href="#">getHumanTime()</a>	String	Returns a string representation of path driving time with measurement units.
<a href="#">getJamsTime()</a>	Integer	Returns the path driving time in seconds, with consideration for traffic.
<a href="#">getLength()</a>	Number	Returns path length in meters.
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">getSegments()</a>	<a href="#">router.Segment</a> []	Returns path segments.
<a href="#">getTime()</a>	Integer	Returns the path driving time in seconds.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IChildOnMap</a> .

## Methods details

### getHumanJamsTime

```
{String} getHumanJamsTime()
```

**Returns** a string representation of path driving time with measurement units, with consideration for traffic.

### getHumanLength

```
{String} getHumanLength()
```

**Returns** a string representation of path length with measurement units.

### getHumanTime

```
{String} getHumanTime()
```

**Returns** a string representation of path driving time with measurement units.

### getJamsTime

```
{Integer} getJamsTime()
```

**Returns** the path driving time in seconds, with consideration for traffic.

### getLength

```
{Number} getLength()
```

**Returns** path length in meters.

### getSegments

```
{router.Segment[]} getSegments()
```

**Returns** path segments.

### getTime

```
{Integer} getTime()
```

**Returns** the path driving time in seconds.

## router.Route

**Note:** The constructor of the `router.Route` class is hidden, as this class is not intended for autonomous initialization.

Extends [IGeoObject](#).

Object that describes the plotted route. The constructor is not available in the `package.full` (a standard set of modules). This module is loaded on demand.

See [route](#)

[Fields](#) | [Events](#) | [Methods](#)

**Fields**

Name	Type	Description
<a href="#">editor</a>	<a href="#">router.Editor</a>	Route editor.
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">IGeoObject</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">properties</a>	<a href="#">IDataManager</a>	Geo object data. Inherited from <a href="#">IGeoObject</a> .
<a href="#">state</a>	<a href="#">IDataManager</a>	State of the geo object. Inherited from <a href="#">IGeoObject</a> .

**Events**

Name	Description
<a href="#">boundsapply</a>	Event for applying the route boundaries to the map with the <code>options.mapStateAutoApply</code> option set.
<a href="#">click</a>	Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">contextmenu</a>	Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">dblclick</a>	Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .
<a href="#">geometrychange</a>	Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li><code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the geometry.</li> </ul> Inherited from <a href="#">IGeoObject</a> .
<a href="#">mapchange</a>	Map reference changed. Data fields: <ul style="list-style-type: none"> <li><code>oldMap</code> - Old map.</li> <li><code>newMap</code> - New map.</li> </ul> Inherited from <a href="#">IParentOnMap</a> .
<a href="#">mousedown</a>	Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a> . Inherited from <a href="#">IDomEventEmitter</a> .

Name	Description
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>

Name	Description
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>overlay: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>oldOverlay: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>oldParent - Old parent.</li> <li>newParent - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>originalEvent: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">update</a>	<p>Route updating event when the route editor is enabled.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getHumanJamsTime()</a>	String	Returns a string representation of route driving time with measurement units, with consideration for traffic.
<a href="#">getHumanLength()</a>	String	Returns a string representation of route length with measurement units.
<a href="#">getHumanTime()</a>	String	Returns a string representation of route driving time with measurement units.
<a href="#">getJamsTime()</a>	Integer	Returns the route driving time in seconds, with consideration for traffic.
<a href="#">getLength()</a>	Number	Returns the length of the route in meters.
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .

Name	Returns	Description
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.  Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay.  Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set.  Inherited from <a href="#">IChildOnMap</a> .
<a href="#">getPaths()</a>	<a href="#">GeoObjectCollection</a>	Returns a collection of paths that make up the route.
<a href="#">getTime()</a>	Integer	Returns the route driving time in seconds.
<a href="#">getViaPoints()</a>	<a href="#">GeoObjectCollection</a>	Returns a collection of throughpoints on the route.
<a href="#">getWayPoints()</a>	<a href="#">GeoObjectCollection</a>	Returns a collection of waypoints on the route.
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .

## Fields details

### editor

```
{router.Editor} editor
```

Route editor.

### Example:

```
// Start of route editing.
ymaps.route(['Moscow', 'Petersburg'], function (route) {
  route.editor.start();
  // ...
  // End of route editing.
  route.editor.stop();
});
```

## Events details

### boundsapply

Event for applying the route boundaries to the map with the options.mapStateAutoApply option set.

## update

Route updating event when the route editor is enabled.

## Methods details

### getHumanJamsTime

```
{String} getHumanJamsTime()
```

**Returns** a string representation of route driving time with measurement units, with consideration for traffic.

### getHumanLength

```
{String} getHumanLength()
```

**Returns** a string representation of route length with measurement units.

### getHumanTime

```
{String} getHumanTime()
```

**Returns** a string representation of route driving time with measurement units.

### getJamsTime

```
{Integer} getJamsTime()
```

**Returns** the route driving time in seconds, with consideration for traffic.

### getLength

```
{Number} getLength()
```

**Returns** the length of the route in meters.

### getPaths

```
{GeoObjectCollection} getPaths()
```

**Returns** a collection of paths that make up the route.

### getTime

```
{Integer} getTime()
```

**Returns** the route driving time in seconds.

### getViaPoints

```
{GeoObjectCollection} getViaPoints()
```

**Returns** a collection of throughpoints on the route.

### getWayPoints

```
{GeoObjectCollection} getWayPoints()
```

**Returns** a collection of waypoints on the route.

## router.Segment

**Note:** The constructor of the router.Segment class is hidden, as this class is not intended for autonomous initialization.

Object that describes a segment of the route. A segment is a part of the route from one maneuver to the next. The constructor is not available in the package.full (a standard set of modules). This module is loaded on demand.

### Methods

#### Methods

Name	Returns	Description
<a href="#">getAction()</a>	String	Returns the direction the route turns at the end of the segment. Possible values: <ul style="list-style-type: none"> <li>left - Left.</li> <li>slight left - Slightly left.</li> <li>hard left - Sharp left turn.</li> <li>right - Right.</li> <li>slight right - Slightly right.</li> <li>hard right - Sharp right turn.</li> <li>none - Straight.</li> <li>back - Turn around.</li> <li>enter roundabout - Entrance to a roundabout intersection.</li> <li>leave roundabout [N] - Exit from a roundabout intersection. The number N is the number of the turn on the circle. This number can be omitted. For example, "leave roundabout" or "leave roundabout 2".</li> <li>merge - Entrance to a highway. Signifies merging with traffic.</li> <li>board ferry - Ferry crossing.</li> </ul>
<a href="#">getAngle()</a>	Number	Checks the angle at which the route turns at the end of the segment.
<a href="#">getCoordinates()</a>	Number[][]	Returns coordinates of a polyline that describes the segment geometry.
<a href="#">getHumanAction()</a>	String	Returns the direction of the turn in the form of a localized human-readable string.
<a href="#">getHumanJamsTime()</a>	String	Returns a string representation of segment driving time with measurement units, with consideration for traffic.

Name	Returns	Description
<a href="#">getHumanLength()</a>	String	Returns a string representation of segment length with measurement units.
<a href="#">getHumanTime()</a>	String	Returns a string representation of segment driving time with measurement units.
<a href="#">getIndex()</a>	Integer	Returns the index of the given segment in the array of all the segments in the path.
<a href="#">getJamsTime()</a>	Integer	Returns the segment driving time in seconds, with consideration for traffic.
<a href="#">getLength()</a>	Number	Returns the length of the segment in meters.
<a href="#">getPolylineEndIndex()</a>	Integer	Returns the index of the point in the path geometry at which the segment ends.
<a href="#">getPolylineStartIndex()</a>	Integer	Returns the index of the point in the path geometry from which the segment begins.
<a href="#">getStreet()</a>	String	Returns the name of the street that the segment of the route goes along.
<a href="#">getTime()</a>	Integer	Returns the segment driving time in seconds.

## Methods details

### getAction

```
{String} getAction()
```

**Returns** the direction the route turns at the end of the segment. Possible values:

- left - Left.
- slight left - Slightly left.
- hard left - Sharp left turn.
- right - Right.
- slight right - Slightly right.
- hard right - Sharp right turn.
- none - Straight.
- back - Turn around.
- enter roundabout - Entrance to a roundabout intersection.
- leave roundabout [N] - Exit from a roundabout intersection. The number N is the number of the turn on the circle. This number can be omitted. For example, "leave roundabout" or "leave roundabout 2".
- merge - Entrance to a highway. Signifies merging with traffic.
- board ferry - Ferry crossing.

**getAngle**

```
{Number} getAngle()
```

Checks the angle at which the route turns at the end of the segment.

**Returns** the angle of the turn (in degrees).

**getCoordinates**

```
{Number[][]} getCoordinates()
```

**Returns** coordinates of a polyline that describes the segment geometry.

**getHumanAction**

```
{String} getHumanAction()
```

**Returns** the direction of the turn in the form of a localized human-readable string.

**getHumanJamsTime**

```
{String} getHumanJamsTime()
```

**Returns** a string representation of segment driving time with measurement units, with consideration for traffic.

**getHumanLength**

```
{String} getHumanLength()
```

**Returns** a string representation of segment length with measurement units.

**getHumanTime**

```
{String} getHumanTime()
```

**Returns** a string representation of segment driving time with measurement units.

**getIndex**

```
{Integer} getIndex()
```

**Returns** the index of the given segment in the array of all the segments in the path.

**getJamsTime**

```
{Integer} getJamsTime()
```

**Returns** the segment driving time in seconds, with consideration for traffic.

**getLength**

```
{Number} getLength()
```

**Returns** the length of the segment in meters.

**getPolylineEndIndex**

```
{Integer} getPolylineEndIndex()
```

**Returns** the index of the point in the path geometry at which the segment ends.

### getPolylineStartIndex

```
{Integer} getPolylineStartIndex()
```

**Returns** the index of the point in the path geometry from which the segment begins.

### getStreet

```
{String} getStreet()
```

**Returns** the name of the street that the segment of the route goes along.

### getTime

```
{Integer} getTime()
```

**Returns** the segment driving time in seconds.

## router.ViaPoint

**Note:** The constructor of the `router.ViaPoint` class is hidden, as this class is not intended for autonomous initialization.

Extends [GeoObject](#).

Object that describes a throughpoint on the route. The constructor is not available in the `package.full` (a standard set of modules). This module is loaded on demand.

[Fields](#) | [Events](#) | [Methods](#)

### Fields

Name	Type	Description
<a href="#">balloon</a>	<a href="#">geoObject.Balloon</a>	Balloon for a geo object. Inherited from <a href="#">GeoObject</a> .
<a href="#">editor</a>	<a href="#">IGeometryEditor</a>	Editor for the geo object geometry. Inherited from <a href="#">GeoObject</a> .
<a href="#">events</a>	<a href="#">event.Manager</a>	Event manager. Inherited from <a href="#">GeoObject</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">GeoObject</a> .
<a href="#">hint</a>	<a href="#">geoObject.Hint</a>	Geo object hint. Inherited from <a href="#">GeoObject</a> .
<a href="#">options</a>	<a href="#">option.Manager</a>	Geo object options manager. Inherited from <a href="#">GeoObject</a> .
<a href="#">properties</a>	<a href="#">data.Manager</a>	Geo object data manager. Inherited from <a href="#">GeoObject</a> .

Name	Type	Description
<a href="#">state</a>	<a href="#">data.Manager</a>	<p>State of the geo object. Defined by the following fields:</p> <ul style="list-style-type: none"> <li>• <code>active</code>: Boolean - Indicates that a balloon is open on the geo object.</li> <li>• <code>hover</code>: Boolean - Indicates that the mouse is currently pointed at the geo object.</li> <li>• <code>drag</code>: Boolean - Indicates that the geo object is being dragged</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>

## Events

Name	Description
<a href="#">balloonclose</a>	<p>Closing the balloon. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">balloonopen</a>	<p>Opening a balloon on a geo object. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">beforedrag</a>	<p>Event preceding the "drag" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>position</code> - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• <code>pixelOffset</code> - Array of two numbers that describe the pixel offset at this step.</li> <li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Names of methods that are accessible via <a href="#">Event.callMethod</a>:</p> <ul style="list-style-type: none"> <li>• <code>setPixelOffset</code> - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent drag event will be canceled.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">beforedragstart</a>	<p>Event preceding the "dragstart" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>position</code> - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">drag</a>	<p>Dragging a geo object. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>position</code> - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• <code>pixelOffset</code> - Array of two numbers that describe the pixel offset at this step.</li> <li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">dragend</a>	<p>End of geo object dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>position</code> - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">dragstart</a>	<p>Start of geo object dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>position</code> - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• <code>domEvent</code> - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">editorstatechange</a>	<p>Change in the state of the editor for the geo object's geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code> - Original event of the geometry editor.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the geometry.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">hintclose</a>	<p>Closing the hint. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">hintopen</a>	<p>Opening a hint on a geo object. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldMap</code> - Old map.</li> <li>• <code>newMap</code> - New map.</li> </ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"><li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li><li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li><li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li></ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>

Name	Description
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	<p>Returns reference to the map.</p> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	<p>Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	<p>The method provides synchronous access to the overlay.</p> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	<p>Returns link to the parent object, or null if the parent element was not set.</p> <p>Inherited from <a href="#">IChildOnMap</a>.</p>
<a href="#">getPathIndex()</a>	Integer	Returns the index of the path that the point is on.
<a href="#">getSegmentIndex()</a>	Integer	Returns the index of the segment of the path where the point is located.

Name	Returns	Description
<code>setParent(parent)</code>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object.  Inherited from <a href="#">IChildOnMap</a> .

## Methods details

### getPathIndex

```
{Integer} getPathIndex()
```

**Returns** the index of the path that the point is on.

### getSegmentIndex

```
{Integer} getSegmentIndex()
```

**Returns** the index of the segment of the path where the point is located.

## router.WayPoint

Extends [GeoObject](#).

Object describing a waypoint on the route.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
router.WayPoint(feature[, options])
```

### Parameters:

Parameter	Default value	Description
<code>feature</code> *	—	Type: Object  Properties and geometry.
<code>options</code>	—	Type: Object  Options.

\* Mandatory parameter/option.

### Fields

Name	Type	Description
<code>balloon</code>	<a href="#">geoObject.Balloon</a>	Balloon for a geo object.  Inherited from <a href="#">GeoObject</a> .
<code>editor</code>	<a href="#">IGeometryEditor</a>	Editor for the geo object geometry.  Inherited from <a href="#">GeoObject</a> .

Name	Type	Description
<a href="#">events</a>	<a href="#">event.Manager</a>	Event manager. Inherited from <a href="#">GeoObject</a> .
<a href="#">geometry</a>	<a href="#">IGeometry</a>  null	Geo object geometry. Inherited from <a href="#">GeoObject</a> .
<a href="#">hint</a>	<a href="#">geoObject.Hint</a>	Geo object hint. Inherited from <a href="#">GeoObject</a> .
<a href="#">options</a>	<a href="#">option.Manager</a>	Geo object options manager. Inherited from <a href="#">GeoObject</a> .
<a href="#">properties</a>	<a href="#">data.Manager</a>	Data manager for a waypoint. If the waypoint was set as an address, the <a href="#">GeocoderMetaData</a> field will contain the geocoder metadata. See <a href="#">geocode</a> .
<a href="#">state</a>	<a href="#">data.Manager</a>	State of the geo object. Defined by the following fields: <ul style="list-style-type: none"> <li>• active: Boolean - Indicates that a balloon is open on the geo object.</li> <li>• hover: Boolean - Indicates that the mouse is currently pointed at the geo object.</li> <li>• drag: Boolean - Indicates that the geo object is being dragged</li> </ul> Inherited from <a href="#">GeoObject</a> .

## Events

Name	Description
<a href="#">balloonclose</a>	Closing the balloon. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">GeoObject</a> .
<a href="#">balloonopen</a>	Opening a balloon on a geo object. Instance of the <a href="#">Event</a> class. Inherited from <a href="#">GeoObject</a> .
<a href="#">beforedrag</a>	Event preceding the "drag" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method: <ul style="list-style-type: none"> <li>• position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• pixelOffset - Array of two numbers that describe the pixel offset at this step.</li> <li>• domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> Names of methods that are accessible via <a href="#">Event.callMethod</a> : <ul style="list-style-type: none"> <li>• setPixelOffset - This method is for correcting the value of the pixel offset that will actually be applied. It takes an argument with the new pixel offset in the form of an array of two numbers.</li> </ul> If the <a href="#">Event.preventDefault</a> method is called for this event, a subsequent drag event will be canceled. Inherited from <a href="#">GeoObject</a> .

Name	Description
<a href="#">beforedragstart</a>	<p>Event preceding the "dragstart" event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>If the <a href="#">Event.preventDefault</a> method is called for this event, any subsequent dragging, as well as the "dragstart" event, will be canceled.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">click</a>	<p>Single left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">contextmenu</a>	<p>Calls the element's context menu. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">dblclick</a>	<p>Double left-click on the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEventManager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">drag</a>	<p>Dragging a geo object. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• pixelOffset - Array of two numbers that describe the pixel offset at this step.</li> <li>• domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">dragend</a>	<p>End of geo object dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">dragstart</a>	<p>Start of geo object dragging. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>• domEvent - Source DOM event (as a <a href="#">DomEvent</a> object), if there is one.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">editorstatechange</a>	<p>Change in the state of the editor for the geo object's geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>• originalEvent - Original event of the geometry editor.</li> </ul> <p>Inherited from <a href="#">GeoObject</a>.</p>

Name	Description
<a href="#">geometrychange</a>	<p>Change to the geo object geometry. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"><li>originalEvent: <a href="#">IEvent</a> - Original event of the geometry.</li></ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">hintclose</a>	<p>Closing the hint. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">hintopen</a>	<p>Opening a hint on a geo object. Instance of the <a href="#">Event</a> class.</p> <p>Inherited from <a href="#">GeoObject</a>.</p>
<a href="#">mapchange</a>	<p>Map reference changed. Data fields:</p> <ul style="list-style-type: none"><li>oldMap - Old map.</li><li>newMap - New map.</li></ul> <p>Inherited from <a href="#">IParentOnMap</a>.</p>
<a href="#">mousedown</a>	<p>Pressing the mouse button over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseenter</a>	<p>Pointing the cursor at the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseleave</a>	<p>Moving the cursor off of the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mousemove</a>	<p>Moving the cursor over the object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">mouseup</a>	<p>Letting go of the mouse button over an object. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchend</a>	<p>End of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <a href="#">IMultiTouchEvent</a> interface.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

Name	Description
<a href="#">multitouchmove</a>	<p>Repeating event during multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">multitouchstart</a>	<p>Start of multitouch. This event is only available on devices that support multitouch. Returns an implementation of the <code>IMultiTouchEvent</code> interface with information about touches. Defines the <code>touches</code> property, which contains a list of touches. Every touch is described by an object that contains the following fields:</p> <ul style="list-style-type: none"> <li>• <code>clientX</code> - X coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>clientY</code> - Y coordinate of the touch relative to the viewable area of the browser.</li> <li>• <code>pageX</code> - X coordinate of the touch relative to the beginning of the document.</li> <li>• <code>pageY</code> - Y coordinate of the touch relative to the beginning of the document.</li> </ul> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">overlaychange</a>	<p>Change to the geo object overlay. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li>• <code>overlay</code>: <a href="#">IOverlay</a> null - Reference to the overlay.</li> <li>• <code>oldOverlay</code>: <a href="#">IOverlay</a> null - Previous overlay of the geo object.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">parentchange</a>	<p>The parent object reference changed.</p> <p>Data fields:</p> <ul style="list-style-type: none"> <li>• <code>oldParent</code> - Old parent.</li> <li>• <code>newParent</code> - New parent.</li> </ul> <p>Inherited from <a href="#">IChild</a>.</p>
<a href="#">propertieschange</a>	<p>Change to the geo object data. Instance of the <code>Event</code> class. Names of fields that are available via the <code>Event.get</code> method:</p> <ul style="list-style-type: none"> <li>• <code>originalEvent</code>: <a href="#">IEvent</a> - Original event of the data manager.</li> </ul> <p>Inherited from <a href="#">IGeoObject</a>.</p>
<a href="#">wheel</a>	<p>Mouse wheel scrolling. When using, keep in mind that mouse events are emulated when a touch screen is used. More information is available in <a href="#">domEvent.manager</a>.</p> <p>Inherited from <a href="#">IDomEventEmitter</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>	Returns reference to the map. Inherited from <a href="#">IParentOnMap</a> .
<a href="#">getOverlay()</a>	<a href="#">vow.Promise</a>	Returns the promise object, which is confirmed by the overlay object at the time it is actually created, or is rejected with an appropriate error message. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getOverlaySync()</a>	<a href="#">IOverlay</a>  null	The method provides synchronous access to the overlay. Inherited from <a href="#">IGeoObject</a> .
<a href="#">getParent()</a>	<a href="#">IParentOnMap</a>  null	Returns link to the parent object, or null if the parent element was not set. Inherited from <a href="#">IChildOnMap</a> .
<a href="#">setParent(parent)</a>	<a href="#">IChildOnMap</a>	Sets the parent object. If the null value is passed, the manager element will only be deleted from the current parent object. Inherited from <a href="#">IChildOnMap</a> .

**Fields details****properties**

```
{data.Manager} properties
```

Data manager for a waypoint. If the waypoint was set as an address, the `GeocoderMetaData` field will contain the geocoder metadata. See [geocode](#).

**shape****shape.Circle**

Extends [IShape](#).

"Circle" pixel shape.

[Constructor](#) | [Methods](#)

**Constructor**

```
shape.Circle(pixelGeometry[, params])
```

Creates the shape.

**Parameters:**

Parameter	Default value	Description
<a href="#">pixelGeometry</a> *	—	Type: <a href="#">IPixelCircleGeometry</a> Pixel geometry of a shape.
<a href="#">params</a>	—	Type: Object Shape parameters.
<a href="#">params.fill</a>	true	Type: Boolean Flag for filling the shape.
<a href="#">params.outline</a>	true	Type: Boolean Flag for an outline.
<a href="#">params.strokeWidth</a>	0	Type: Number The outline width, in pixels.

\* Mandatory parameter/option.

## Methods

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the shape. Inherited from <a href="#">IShape</a> .
<a href="#">equals(shape)</a>	Boolean	Returns true if the passed shape is equivalent to the given one. Inherited from <a href="#">IShape</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area spanning the shape. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values. Inherited from <a href="#">IShape</a> .
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	Returns pixel geometry of a shape. Inherited from <a href="#">IShape</a> .
<a href="#">getType()</a>	String	Returns ID of the shape type. Inherited from <a href="#">IShape</a> .
<a href="#">scale(factor)</a>	<a href="#">IShape</a>	Creates a scaled copy of the shape. Inherited from <a href="#">IShape</a> .

Name	Returns	Description
<a href="#">shift(offset)</a>	<a href="#">IShape</a>	Creates a copy of the shape that is shifted by the specified amount.  Inherited from <a href="#">IShape</a> .

## shape.LineString

Extends [IShape](#).

The "Polyline" pixel shape.

[Constructor](#) | [Methods](#)

### Constructor

```
shape.LineString(pixelGeometry[, params])
```

Creates the shape.

### Parameters:

Parameter	Default value	Description
<a href="#">pixelGeometry</a> *	—	Type: <a href="#">IPixelLineStringGeometry</a>  Pixel geometry of a shape.
<a href="#">params</a>	—	Type: Object  Shape parameters.
<a href="#">params.strokeWidth</a>	1	Type: Number  The line width, in pixels.

\* Mandatory parameter/option.

### Methods

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the shape.  Inherited from <a href="#">IShape</a> .
<a href="#">equals(shape)</a>	Boolean	Returns true if the passed shape is equivalent to the given one.  Inherited from <a href="#">IShape</a> .

Name	Returns	Description
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area spanning the shape. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IShape</a> .
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	Returns pixel geometry of a shape.  Inherited from <a href="#">IShape</a> .
<a href="#">getType()</a>	String	Returns ID of the shape type.  Inherited from <a href="#">IShape</a> .
<a href="#">scale(factor)</a>	<a href="#">IShape</a>	Creates a scaled copy of the shape.  Inherited from <a href="#">IShape</a> .
<a href="#">shift(offset)</a>	<a href="#">IShape</a>	Creates a copy of the shape that is shifted by the specified amount.  Inherited from <a href="#">IShape</a> .

## shape.MultiGeometry

Extends [IShape](#).

"Multipolygon" pixel shape.

[Constructor](#) | [Methods](#)

### Constructor

```
shape.MultiGeometry(pixelGeometry[, params])
```

Creates the shape.

### Parameters:

Parameter	Default value	Description
<a href="#">pixelGeometry</a> *	—	Type: <a href="#">IPixelMultiGeometry</a>  Pixel geometry of a shape.
<a href="#">params</a>	—	Type: Object  Shape parameters.
<a href="#">params.fill</a>	true	Type: Boolean  Flag for filling the shape.

Parameter	Default value	Description
<a href="#">params.outline</a>	true	Type: Boolean Flag for an outline.
<a href="#">params.strokeWidth</a>	0	Type: Number The outline width, in pixels.

\* Mandatory parameter/option.

## Methods

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the shape.  Inherited from <a href="#">IShape</a> .
<a href="#">equals(shape)</a>	Boolean	Returns true if the passed shape is equivalent to the given one.  Inherited from <a href="#">IShape</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area spanning the shape. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IShape</a> .
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	Returns pixel geometry of a shape.  Inherited from <a href="#">IShape</a> .
<a href="#">getType()</a>	String	Returns ID of the shape type.  Inherited from <a href="#">IShape</a> .
<a href="#">scale(factor)</a>	<a href="#">IShape</a>	Creates a scaled copy of the shape.  Inherited from <a href="#">IShape</a> .
<a href="#">shift(offset)</a>	<a href="#">IShape</a>	Creates a copy of the shape that is shifted by the specified amount.  Inherited from <a href="#">IShape</a> .

## shape.MultiPolygon

Extends [IShape](#).

"Multipolygon" pixel shape.

[Constructor](#) | [Methods](#)

**Constructor**

```
shape.MultiPolygon(pixelGeometry[, params])
```

Creates the shape.

**Parameters:**

Parameter	Default value	Description
<a href="#">pixelGeometry</a> *	—	Type: <a href="#">IPixelMultiPolygonGeometry</a> Pixel geometry of a shape.
<a href="#">params</a>	—	Type: Object Shape parameters.
<a href="#">params.fill</a>	true	Type: Boolean Flag for filling the shape.
<a href="#">params.outline</a>	true	Type: Boolean Flag for an outline.
<a href="#">params.strokeWidth</a>	0	Type: Number The outline width, in pixels.

\* Mandatory parameter/option.

**Methods**

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the shape.  Inherited from <a href="#">IShape</a> .
<a href="#">equals(shape)</a>	Boolean	Returns true if the passed shape is equivalent to the given one.  Inherited from <a href="#">IShape</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area spanning the shape. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IShape</a> .
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	Returns pixel geometry of a shape.  Inherited from <a href="#">IShape</a> .

Name	Returns	Description
<a href="#">getType()</a>	String	Returns ID of the shape type. Inherited from <a href="#">IShape</a> .
<a href="#">scale(factor)</a>	<a href="#">IShape</a>	Creates a scaled copy of the shape. Inherited from <a href="#">IShape</a> .
<a href="#">shift(offset)</a>	<a href="#">IShape</a>	Creates a copy of the shape that is shifted by the specified amount. Inherited from <a href="#">IShape</a> .

## shape.Polygon

Extends [IShape](#).

"Polygon" pixel shape.

[Constructor](#) | [Methods](#)

### Constructor

```
shape.Polygon(pixelGeometry[, params])
```

Creates the shape.

### Parameters:

Parameter	Default value	Description
<a href="#">pixelGeometry</a> *	—	Type: <a href="#">IPixelPolygonGeometry</a> Pixel geometry of a shape.
<a href="#">params</a>	—	Type: Object Shape parameters.
<a href="#">params.fill</a>	true	Type: Boolean Flag for filling the shape.
<a href="#">params.outline</a>	true	Type: Boolean Flag for an outline.
<a href="#">params.strokeWidth</a>	0	Type: Number The outline width, in pixels.

\* Mandatory parameter/option.

### Methods

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the shape. Inherited from <a href="#">IShape</a> .

Name	Returns	Description
<a href="#">equals(shape)</a>	Boolean	Returns true if the passed shape is equivalent to the given one.  Inherited from <a href="#">IShape</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area spanning the shape. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IShape</a> .
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	Returns pixel geometry of a shape.  Inherited from <a href="#">IShape</a> .
<a href="#">getType()</a>	String	Returns ID of the shape type.  Inherited from <a href="#">IShape</a> .
<a href="#">scale(factor)</a>	<a href="#">IShape</a>	Creates a scaled copy of the shape.  Inherited from <a href="#">IShape</a> .
<a href="#">shift(offset)</a>	<a href="#">IShape</a>	Creates a copy of the shape that is shifted by the specified amount.  Inherited from <a href="#">IShape</a> .

## shape.Rectangle

Extends [IShape](#).

"Rectangle" pixel shape.

[Constructor](#) | [Methods](#)

### Constructor

```
shape.Rectangle(pixelGeometry[, params])
```

Creates the shape.

### Parameters:

Parameter	Default value	Description
<a href="#">pixelGeometry</a> *	—	Type: <a href="#">IPixelRectangleGeometry</a>  Pixel geometry of a shape.
<a href="#">params</a>	—	Type: Object  Shape parameters.

Parameter	Default value	Description
<a href="#">params.fill</a>	true	Type: Boolean Flag for filling the shape.
<a href="#">params.outline</a>	true	Type: Boolean Flag for an outline.
<a href="#">params.strokeWidth</a>	0	Type: Number The outline width, in pixels.

\* Mandatory parameter/option.

## Methods

Name	Returns	Description
<a href="#">contains(position)</a>	Boolean	Checks whether the passed point is located inside the shape.  Inherited from <a href="#">IShape</a> .
<a href="#">equals(shape)</a>	Boolean	Returns true if the passed shape is equivalent to the given one.  Inherited from <a href="#">IShape</a> .
<a href="#">getBounds()</a>	Number[][] null	Returns coordinates of the two opposite corners of the area spanning the shape. The first item in the array is the corner with the smallest coordinate values relative to the rest of the points in the area; the second item is the corner with the largest coordinate values.  Inherited from <a href="#">IShape</a> .
<a href="#">getGeometry()</a>	<a href="#">IPixelGeometry</a>	Returns pixel geometry of a shape.  Inherited from <a href="#">IShape</a> .
<a href="#">getType()</a>	String	Returns ID of the shape type.  Inherited from <a href="#">IShape</a> .
<a href="#">scale(factor)</a>	<a href="#">IShape</a>	Creates a scaled copy of the shape.  Inherited from <a href="#">IShape</a> .
<a href="#">shift(offset)</a>	<a href="#">IShape</a>	Creates a copy of the shape that is shifted by the specified amount.  Inherited from <a href="#">IShape</a> .

## shape.storage

Static object.

Instance of [util.Storage](#)

Storage for hotspot shape geometries.

[Methods](#)

### Methods

Name	Returns	Description
<a href="#">add(key, object)</a>	<a href="#">util.Storage</a>	Adds an object to storage.
<a href="#">get(key)</a>	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
<a href="#">remove(key)</a>	<a href="#">util.Storage</a>	Deletes the "key: value" pair from storage.

## suggest

Static function.

Processes requests for search suggestions. Returns a promise object that is either rejected with an error, or confirmed by an array of objects in the format { displayName: "Mitishi, Moscow region", value: "Russia, Moscow region, Mitishi " }. The displayName field represents the toponym in a user-friendly way, and the value field represents the value which should be inserted into the search field after the user selects the suggestion.

**Returns** a Promise object.

```
{ vow.Promise } suggest(request[, options])
```

### Parameters:

Parameter	Default value	Description
<a href="#">request</a> *	—	Type: String Request string.
<a href="#">options</a>	—	Type: Object Options.
<a href="#">options.boundedBy</a>	—	Type: Number[][] A rectangular area on the map, where the object being searched for is presumably located. Must be set as an array, such as [[30, 40], [50, 50]].
<a href="#">options.provider</a>	'yandex#map'	Type: <a href="#">ISuggestProvider</a>  String Search suggestion provider. You can use the 'yandex#map' built-in search suggestion provider for map objects, or specify your own.

Parameter	Default value	Description
<code>options.results</code>	—	Type: Number  Maximum number of results to be returned.

\* Mandatory parameter/option.

#### Example:

```
ymaps.suggest('myt').then(function (items) {
  // items - Array of search suggestions.
});
```

## SuggestView

Extends [ICustomizable](#), [IEventEmitter](#).

Creates a drop-down list with search suggestions and attaches it to the HTML element `<input type="text">`.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
SuggestView(element[, options])
```

#### Parameters:

Parameter	Default value	Description
<code>element *</code>	—	Type: HTMLElement string  HTML element or its ID.
<code>options</code>	—	Type: Object  Options.
<code>options.boundedBy</code>	—	Type: Number[][]  A rectangular area on the map, where the object being searched for is presumably located. Must be set as an array, such as <code>[[30, 40], [50, 50]]</code> .
<code>options.container</code>	—	Type: HTMLElement  HTML element for placing the display of the suggestions panel. If omitted, the suggestions panel is added to the parent of the HTML "input" element that it is created for.

Parameter	Default value	Description
<a href="#">options.layout</a>	'islands#suggestView'	<p>Type: <a href="#">String</a>  <a href="#">ISuggestViewLayout</a></p> <p>Panel layout.</p> <p>The layout constructor is passed an object containing the fields:</p> <ul style="list-style-type: none"> <li>• <code>suggestView</code> - Link to the panel with search suggestions.</li> <li>• <code>options</code> - Options manager for the control <code>suggestView.options</code>.</li> <li>• <code>state</code> - State manager for the control <code>suggestView.state</code>.</li> </ul> <p>The layout adapts its appearance based on the state and options of the suggestions panel. The control, in turn, reacts to layout interface events and changes the values of fields for <code>suggestView.state</code> depending on the commands received.</p>
<a href="#">options.offset</a>	—	<p>Type: <a href="#">Number</a>[]</p> <p>Offsets of the suggestions panel from its default location (by default, the suggestions panel is attached to the lower edge of the "input" element and has the same width as it). Set as horizontal and vertical offsets relative to the lower-left corner of the "input" element.</p>
<a href="#">options.provider</a>	"yandex#map"	<p>Type: <a href="#">String</a>  <a href="#">ISuggestProvider</a></p> <p>Provider for search suggestions. May be set as an object implementing the <a href="#">ISuggestProvider</a> interface, or the standard value "yandex#map".</p>
<a href="#">options.results</a>	5	<p>Type: <a href="#">Number</a></p> <p>Maximum quantity of suggestions to display.</p>
<a href="#">options.width</a>	—	<p>Type: <a href="#">Number</a></p> <p>Width of the suggestions panel. By default, the same as the width of the HTML "input" element to which the panel is attached.</p>
<a href="#">options.zIndex</a>	40000	<p>Type: <a href="#">Number</a></p> <p>The z-index for the dom element of the suggestions panel.</p>

\* Mandatory parameter/option.

#### Examples:

1.

```
<input type="text" id="suggest"/>
<script src="//api-maps.yandex.ru/2.1/?lang=ru_RU&amp;load=SuggestView&amp;onload=onLoad">&lt;/script&gt;
```

```
<script>
function onLoad (ymaps) {
  var suggestView = new ymaps.SuggestView('suggest');
}
</script>
```

**2.**

```
<input type="text" id="suggest"/>
<script src="//api-maps.yandex.ru/2.1/?lang=ru_RU&load=SuggestView&onload=onLoad"></script>
<script>
function onLoad (ymaps) {
  var suggestView = new ymaps.SuggestView('suggest', {results: 1, offset: [20, 30]});
}
</script>
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">state</a>	<a href="#">data.Manager</a>	State of the search suggestions panel. Names of fields that are available via the <a href="#">data.Manager.get</a> method: <ul style="list-style-type: none"> <li><code>request</code>: String — Current active request.</li> <li><code>items</code>: Object[] - Array of search suggestions (objects with the <code>value</code> and <code>displayName</code> fields).</li> <li><code>activeIndex</code>: Number null - Index of the currently active suggestion selected by the user with the mouse or keyboard, or null, if none of the suggestions is active.</li> <li><code>panelClosed</code>: Boolean - Indicates whether the user closed the panel by pressing ESC or choosing one of the suggestions.</li> </ul>

**Events**

Name	Description
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .
<a href="#">select</a>	The user selected one of the search suggestions. Names of fields that are available via the <a href="#">Event.get</a> method <ul style="list-style-type: none"> <li><code>item</code> - A search suggestion (an object with the <code>displayName</code> and <code>value</code> fields).</li> </ul>

**Methods**

Name	Description
<a href="#">destroy()</a>	Destroys the search suggestions panel.

**Fields details****state**

```
{data.Manager} state
```

State of the search suggestions panel. Names of fields that are available via the [data.Manager.get](#) method:

- `request`: String — Current active request.
- `items`: Object[] - Array of search suggestions (objects with the `value` and `displayName` fields).
- `activeIndex`: Number|null - Index of the currently active suggestion selected by the user with the mouse or keyboard, or null, if none of the suggestions is active.
- `panelClosed`: Boolean - Indicates whether the user closed the panel by pressing ESC or choosing one of the suggestions.

## Events details

### select

The user selected one of the search suggestions. Names of fields that are available via the [Event.get](#) method

- `item` – A search suggestion (an object with the "displayName" and "value" fields).

## Methods details

### destroy

```
{ } destroy()
```

Destroys the search suggestions panel.

## template

### template.filtersStorage

Static object.

Instance of [util.Storage](#)

Storage for template filters. Filters from storage can be used in all layouts created using [templateLayoutFactory](#). By default, the storage declares the following keys for filters:

- `default` — Allows setting default values. For example, like this: `{{ properties.header|default:"Title" }}`

### Methods

#### Examples:

##### 1.

```
// Writing a simple filter that will convert a date
// from the format "dd.mm.yyyy" to the format "dd month yyyy".
// To do this, we need to create a filter function that will return the new value.

// When the filter is called, the following arguments are passed to the filter:
// the template data manager data.Manager, the value, and the value set for the filter.
var dateFilter = function (data, dateString, filterValue) {
    var months = [
        'january', 'february', 'march', 'april',
        'may', 'june', 'july', 'august',
        'september', 'october', 'november', 'december'
    ];
    var date = dateString.split('.');

    date[1] = months[parseInt(date[1], 10)];
    return date.join(' ');
};

ymaps.template.filtersStorage.add('date', dateFilter);

// Then we can use it in constructions like
// {{ "21.10.2014"|date }} the value will be "21 october 2014".
```

##### 2.

```
// Writing a filter that will find and replace substrings in text.
// The format for substitution values in the filter is "subString_newSubString".
```

```
// When the filter is called, the following arguments are passed to the function:
// the template data manager data.Manager, the text, and the value set for the filter.
var replaceFilter = function (data, text, replace) {
  replace = replace.trim();
  // Removing quotation marks.
  replace = replace.slice(1, replace.length - 1);

  // Finding the part that comes before "_" in the text and replacing it with what comes after it.
  var values = replace.split('_');
  var from = new RegExp(values[0], 'g');
  var to = values[1];

  return text.replace(from, to);
};

// Now we can use this in templates of constructions like
// {{ "text test replace"|replace: "test_replaced test" }} the value will be "text replaced test replace".
```

### 3.

```
// In this example, the value of the "colorClass" option and the value of the "header" property are added to the
// layout.
// If the "header" property doesn't have a value, the string "Title" is inserted.
var LayoutClass = ymaps.templateLayoutFactory.createClass(
  '<h1 class="{ options.colorClass }">'+
  '{{ properties.header|default:"Title" }}'+
  '</h1>';
);
```

## Methods

Name	Returns	Description
<a href="#">add(key, object)</a>	<a href="#">util.Storage</a>	Adds an object to storage.
<a href="#">get(key)</a>	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
<a href="#">remove(key)</a>	<a href="#">util.Storage</a>	Deletes the "key: value" pair from storage.

## Template

Templating engine. The Yandex.Maps API supports the base syntax for the Twig/Django Templates languages. The following operations are supported:

- Substitution value - `{{ field_name }}`.
- If the requested data field is missing or has an empty value you can provide the default value - `{{ field_name|default:default_value }}` The default value can be a number, a string (in quotes) or a different data field.
- By default the value is processed by the escape function to prevent XSS vulnerabilities. To undo this behavior, add the **"raw"** filter - `{{ field_name|default: default_value|raw }}`.
- Using the [template.filtersStorage](#) you can create your own filters and use them similarly to the described above.
- To insert a sub layout, use a construction like `{% include field_name_or_key %}`. The templating engine when it finds such a construction, will try to use the value in the field as a key of the nested layout.
- The condition is written as:
 

```
{% if condition %} ... {% else %} ... {% endif %}
```

 or you can omit the **else** or **elseif** block. You can use any constructions of the template language inside **if**, **else** and **elseif** blocks.
- Use the **for** construction to iterate an array or an object.
 

```
{% for value in array_or_hash %} ... {% endfor %}
```

. You can use any constructions of the template language inside the **for** block.
- To obtain the iteration index in the array or the field name in the hash, use the following construction:
 

```
{% for key, value in array_or_hash %} ... {% endfor %}
```

.

[Constructor](#) | [Methods](#)

## Constructor

`Template(text)`

### Parameters:

Parameter	Default value	Description
<code>text</code> *	—	Type: String Template string

\* Mandatory parameter/option.

### Examples:

#### 1.

```
// Getting the user name from the data manager data.Manager.
// If the name is not specified, the result string will be «Unregistered user».
var data = new ymaps.data.Manager({
  user: {
    name: "Viktor",
    age: 25
  },
  home: [55.72725771214265, 37.640390506634006]
});

var template = new ymaps.Template('{{ user.name |default: "Unregistered user"}}');
var result = template.build(data);
console.log(result.text); // Outputs «Viktor» to the console.
```

#### 2.

```
// Let's assume we have 3 user groups and we need to print an individual greeting for each group.
var data = new ymaps.data.Manager({
  groups: {
    administrator: {
      id: 1,
      name: "administrator"
    },
    moderator: {
      id: 2,
      name: "moderator"
    },
    user: {
      id: 3,
      name: "user"
    }
  },
  userGroupId: 2
});
var template = new ymaps.Template('Hi, \
{% if (userGroupId == 1) %}{{ groups.administrator.name }}\
{% elseif (userGroupId == 2) %}{{ groups.moderator.name }}\
{% elseif (userGroupId == 3) %}{{ groups.user.name }}\
{% else %}guest{% endif %}!');

var result = template.build(data);
console.log(result.text); // Outputs «Hi, moderator!» to the console.
```

## Methods

Name	Returns	Description
<code>build(data)</code>	Object	Returns object with following fields: <ul style="list-style-type: none"> <li><code>{String} text</code> — the result of templating.</li> <li><code>{Object[]} renderedValues</code> - an array with the used data from the manager.</li> </ul>

## Methods details

### build

```
{Object} build(data)
```

Returns object with following fields:

- {String} text — the result of templating.
- {Object[]} renderedValues - an array with the used data from the manager.

Parameters:

Parameter	Default value	Description
<code>data</code> *	—	Type: <a href="#">IDataManager</a> Data manager.

\* Mandatory parameter/option.

### Example:

```
// Let's get the house address from the existing coordinates and output (according to the template)
// all its inhabitants in the format: «name: age».
var data = new ymaps.data.Manager({
  users: [
    {name: "Vitaly", age: 40},
    {name: "George", age: 20}
  ],
  home: {
    coords: [55.736652, 37.620589],
    address: null
  }
});
var template = new ymaps.Template('{{home.address}}: <ul>{% for user in users %}<li>{{user.name}}: {{user.age}}</li>{% endfor %}</ul>');

// Let's do the reverse geocoding using geocode.
ymaps.geocode(data.get('home.coords')).then(function (res) {
  var address = res.geoObjects.get(0).properties.get('name');
  // Setting the obtained address to the manager.
  data.set('home.address', address);

  // Completing the template with the obtained data.
  var result = template.build(data);
  // Output the result to the console.
  console.log(result.text);
});
```

## templateLayoutFactory

Static object.

Factory for creating a layout class from a text template. Allows creating classes that implement the interface [ILayout](#) using a template language. The Yandex.Maps API supports the base syntax for the Twig/Django Templates languages. For more information about the syntax, see the description of the [Template](#).

See [layout.templateBased.Base](#)

### Methods

### Examples:

#### 1.

```
// In this example, the value of the "colorClass" option and the value of the "header" property are added to the layout.
// If the "header" property doesn't have a value, the string "Title" is inserted.
var LayoutClass = ymaps.templateLayoutFactory.createClass(
  '<h1 class="{{ options.colorClass }}">{{ properties.header|default:"Title" }}</h1>';
);
```

**2.**

```
// One of the layouts is enabled, depending on the value of the "width" option.
var LayoutClass = ymaps.templateLayoutFactory.createClass(
  '{% if options.width > 200 %}' +
  // The appropriate layout will be found in the options.
  '{% include options.wideLayout %}' +
  '{% else %}' +
  // Writing the key explicitly.
  '{% include "cluster#balloonCarousel" %}' +
  '{% endif %}'
);
```

**3.**

```
// Outputting an array of names to the balloon layout.
var CustomLayoutClass = ymaps.templateLayoutFactory.createClass(
  '<ul>' +
  '{% for name in properties.names %}' +
  // The "name" variable is only visible in the for ... endfor block
  '<li>{{ name }}</li>' +
  '{% endfor %}' +
  '</ul>'
);

var placemark = new ymaps.Placemark([54.83, 37.11], {
  names: ['Logan', 'Sofia', 'Mason', 'Layla']
}, {
  balloonContentLayout: CustomLayoutClass
});
```

**4.**

```
// Getting the names of fields.
var CustomLayoutClass = ymaps.templateLayoutFactory.createClass(
  '<ul>' +
  '{% for key, value in properties.hash %}' +
  '<li>{{ key }} {{ value }}</li>' +
  '{% endfor %}' +
  '</ul>'
);

var placemark = new ymaps.Placemark([54.83, 37.11], {
  hash: { key1: "value1", key2: "value2", key3: "value3" }
}, {
  balloonContentLayout: CustomLayoutClass
});
```

**Methods**

Name	Static	Returns	Description
<code>templateLayoutFactory.createClass(<a href="#">template</a>[, <a href="#">overrides</a>[, <a href="#">staticMethods</a>]])</code>		Function	Returns layout constructor. The created class inherits from the class <a href="#">layout.templateBased.Base</a> with a redefined list of methods specified in <code>overrides</code> .

**Methods details****createClass**

```
{Function} <static> templateLayoutFactory.createClass(template[, overrides[, staticMethods]])
```

**Returns** layout constructor. The created class inherits from the class [layout.templateBased.Base](#) with a redefined list of methods specified in `overrides`.

**Parameters:**

Parameter	Default value	Description
<a href="#">template</a> *	—	Type: String  Template for HTML content for layouts.
<a href="#">overrides</a>	—	Type: Object  Redefining parent methods. The build, clear and rebuild methods can be redefined or expanded.
<a href="#">staticMethods</a>	—	Type: Object  Setting static methods for classes.

\* Mandatory parameter/option.

## traffic

### traffic.provider

#### traffic.provider.Actual

Extends [ITrafficProvider](#).

Provider of real-time traffic data. Accessible in the provider storage by the key 'traffic#actual'.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

#### Constructor

```
traffic.provider.Actual([options[, state]])
```

Creates a provider of real-time traffic data.

#### Parameters:

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object  Provider options. Options for provider layers are set via the options for the global collection of layers, <a href="#">Map.layers</a> . <ul style="list-style-type: none"> <li>Options for the image layer <a href="#">Layer</a> are defined with the 'trafficImage' prefix.</li> <li>Options for the hotspot layer <a href="#">hotspot.Layer</a> are defined with the 'trafficJam' prefix.</li> <li>Options for the infopoint layer are defined with the 'trafficInfo' prefix. The infopoint layer is an instance of the <a href="#">hotspot.Layer</a> class.</li> </ul>

Parameter	Default value	Description
<a href="#">options.autoUpdate</a>	true	Type: Boolean  Flag that enables automatically updating traffic data. Automatic updates occur only when the "mousemove" event occurs every 4 minutes on the map. If this event does not occur, traffic stops being updated until there is a new event.
<a href="#">state</a>	—	Type: Object  Provider state.
<a href="#">state.infoLayerShown</a>	false	Type: Boolean  Flag that enables displaying the traffic events layer.

**Example:**

```
// Creating a provider for current traffic with the traffic events layer enabled
// and putting it on the map.
var actualProvider = new ymaps.traffic.provider.Actual({}, {infoLayerShown: true});
actualProvider.setMap(myMap);

// Forbidding showing balloons for clicks on the infopoint layer.
myMap.layers.options.set({
  // The option name is formed by adding the 'trafficInfo' prefix
  // to the hotspot layer option 'openBalloonOnClick'.
  trafficInfoOpenBalloonOnClick: false
});

// ...
// Deleting the provider from the map.
actualProvider.setMap(null);
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager.  Inherited from <a href="#">ICustomizable</a> .
<a href="#">state</a>	<a href="#">data.Manager</a>	Provider state. Names of fields that are available via the <a href="#">data.Manager.get</a> method: <ul style="list-style-type: none"> <li><code>isInited</code> - Flag for whether the provider is ready to provide data.</li> <li><code>infoLayerShown</code> - Flag for whether the traffic events layer is shown.</li> <li><code>timestamp</code> - Current time in the UTC+0 time zone, in seconds.</li> <li><code>localtime</code> - Local time that the server is currently sending data for, in the format HH:MM.</li> <li><code>level</code> - Traffic level in points from 0 to 10.</li> <li><code>isotime</code> - String containing the current date in the format "YYYY-MM-DDThh:mm:ss±hhmm".</li> </ul>

**Events**

Name	Description
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .

**Methods**

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>  null	Returns reference to the map. Inherited from <a href="#">ITrafficProvider</a> .
<a href="#">setMap(Reference)</a>		
<a href="#">update()</a>		Sends a request to update traffic.

**Fields details****state**

```
{data.Manager} state
```

Provider state. Names of fields that are available via the `data.Manager.get` method:

- `isInited` - Flag for whether the provider is ready to provide data.
- `infoLayerShown` - Flag for whether the traffic events layer is shown.
- `timestamp` - Current time in the UTC+0 time zone, in seconds.
- `localtime` - Local time that the server is currently sending data for, in the format HH:MM.
- `level` - Traffic level in points from 0 to 10.
- `isotime` - String containing the current date in the format "YYYY-MM-DDThh:mm:ss±hhmm".

**Example:**

```
var actualProvider = new ymaps.traffic.provider.Actual();
actualProvider.setMap(myMap);
actualProvider.state.events.add('change', function () {
  if (actualProvider.state.get('isInited')) {
    alert('The provider is ready to provide data.');
```

**Methods details****update**

```
{ } update ()
```

Sends a request to update traffic.

**Example:**

```
var trafficControl = new ymaps.control.TrafficControl({shown: true});
map.controls.add(trafficControl);
function updateProvider () {
  trafficControl.getProvider('traffic#actual').update();
}
// Sending a request to update data every 4 minutes.
window.setInterval(updateProvider, 4 * 60 * 1000);
```

**traffic.provider.Archive**

Extends [ITrafficProvider](#).

Provider for the traffic archive. This lets us show the normal state of traffic for a given region on a particular day of the week and at a particular time.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
traffic.provider.Archive([options[, state]])
```

Creates an instance of the traffic archive provider.

#### Parameters:

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object  Provider options. Options for provider layers are set via the options for the global collection of layers, <a href="#">Map.layers</a> . <ul style="list-style-type: none"> <li>Options for the image layer <a href="#">Layer</a> are defined with the 'trafficImage' prefix.</li> <li>Options for the hotspot layer <a href="#">hotspot.Layer</a> are defined with the 'trafficJam' prefix.</li> </ul>
<a href="#">options.showCurrentTimeFirst</a>	true	Type: Boolean  When archived data is first shown, set a time near the current time.
<a href="#">state</a>	—	Type: Object  Provider state.
<a href="#">state.timestamp</a>	—	Type: Number  The time that "normal" traffic is being shown for. This is the time from Monday at 00:00 to the desired time, in seconds. It must be a multiple of $60 * 15 = 900$ , since data on the server is available for times with a difference of 15 minutes. The time is set for the universal time zone (UTC+0).

#### Example:

```
// Creating a provider for "normal" traffic and giving it a timestamp of 17:47 on Wednesday
// in the universal time zone. Note that the local time depends on
// the location of the map center.
// For example, 17:47 in the universal time zone is 21:47 in Moscow.

// Calculating the value of the timestamp parameter for the desired time.
var timestamp = 2 * 24 * 60 * 60 + // twice every 24 hours - this is the time for Monday and Tuesday
    17 * 60 * 60 + // 17 hours have passed since 00:00 Wednesday
    45 * 60; // since the time must be in 15-minute increments, use 45 instead of 47.
var archiveProvider = new ymaps.traffic.provider.Archive({
    // Don't display a time near the current time on the first opening
    showCurrentTimeFirst: false
}, {
    // Setting the starting time independently.
    timestamp: timestamp
});
archiveProvider.setMap(map);

// Don't show popup hints for the traffic layer.
myMap.layers.options.set({
    // The option name is formed by adding the 'trafficJam' prefix
    // to the hotspot layer option 'openHintOnHover'.
```

```

    trafficJamOpenHintOnHover: false
  });

  // ...
  // Removing the provider from the map.
  archiveProvider.setMap(null);

```

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager. Inherited from <a href="#">IEventEmitter</a> .
<a href="#">options</a>	<a href="#">IOptionManager</a>	Options manager. Inherited from <a href="#">ICustomizable</a> .
<a href="#">state</a>	<a href="#">data.Manager</a>	Provider state. Names of fields that are available via the <a href="#">data.Manager.get</a> method: <ul style="list-style-type: none"> <li><code>isInited</code> - Flag for whether the provider is ready to provide data.</li> <li><code>timeZone</code> - Time offset for the current time zone relative to UTC+0. Measured in seconds.</li> <li><code>dst</code> - Flag for switching to winter/summer time (daylight saving time). When <code>dst='dst'</code> it is summer time.</li> <li><code>timestamp</code> - Current time in the UTC+0 time zone, in seconds.</li> <li><code>localtime</code> - The local time that the server returns in the response.</li> <li><code>level</code> - Traffic level in points from 0 to 10.</li> </ul>

## Events

Name	Description
<a href="#">optionschange</a>	Change to the object options. Inherited from <a href="#">ICustomizable</a> .

## Methods

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>  null	Returns reference to the map. Inherited from <a href="#">ITrafficProvider</a> .
<a href="#">getTime()</a>	Object null	Returns the day of the week, hour, and minutes for the provider's status with consideration of the time zone and adjustments for daylight saving time. In other words, this is the time the user sees in the traffic control.
<a href="#">setMap(Reference)</a>		

Name	Returns	Description
<code>setTime(time[, callback])</code>		<p>Allows to set the time for an archive provider in minutes, hours and days. Sets the local time only after the provider initializes the "timeZone" and "dst" fields.</p> <ul style="list-style-type: none"> <li>timeZone - Field that displays which time zone the map center is currently located in. When the map center is moved from one time zone to another, the local time may change.</li> <li>dst - Flag for switching to summer/winter time (daylight saving time). When dst='dst' it indicates summer time.</li> </ul> <p>The "timestamp" field serves as a permanent part of the time for providers of "normal" traffic, and reflects the current time in the zero time zone (UTC+0). When changing from one time zone to another, "timestamp" does not change. You can get the values of the "timestamp", "dst", and "timeZone" fields from the <a href="#">traffic.provider.Archive.state</a> field.</p>

## Fields details

### state

```
{data.Manager} state
```

Provider state. Names of fields that are available via the `data.Manager.get` method:

- isInited - Flag for whether the provider is ready to provide data.
- timeZone - Time offset for the current time zone relative to UTC+0. Measured in seconds.
- dst - Flag for switching to winter/summer time (daylight saving time). When dst='dst' it is summer time.
- timestamp - Current time in the UTC+0 time zone, in seconds.
- localtime - The local time that the server returns in the response.
- level - Traffic level in points from 0 to 10.

### Example:

```
var archiveProvider = new ymaps.traffic.provider.Archive();
archiveProvider.setMap(myMap);
archiveProvider.state.events.add('change', function () {
  if (archiveProvider.state.get('isInited')) {
    alert('Provider is ready to provide data.');

```

## Methods details

### getTime

```
{Object|null} getTime()
```

Returns the day of the week, hour, and minutes for the provider's status with consideration of the time zone and adjustments for daylight saving time. In other words, this is the time the user sees in the traffic control.

**Returns** object with fields

- `dayOfWeek` - Abbreviations of days of the week. 'mon', 'tue', 'wed', 'thu', 'fri', 'sat', 'sun'.
- `hours` - Hours.
- `minutes` - Minutes.

If the map center is located at a point that we cannot determine the time zone for, the function returns null; if we don't know which time zone we are in, we can't find out the local time.

### setTime

```
{ } setTime(time[, callback])
```

Allows to set the time for an archive provider in minutes, hours and days. Sets the local time only after the provider initializes the "timeZone" and "dst" fields.

- `timeZone` - Field that displays which time zone the map center is currently located in. When the map center is moved from one time zone to another, the local time may change.
- `dst` - Flag for switching to summer/winter time (daylight saving time). When `dst='dst'` it indicates summer time.

The "timestamp" field serves as a permanent part of the time for providers of "normal" traffic, and reflects the current time in the zero time zone (UTC+0). When changing from one time zone to another, "timestamp" does not change. You can get the values of the "timestamp", "dst", and "timeZone" fields from the [traffic.provider.Archive.state](#) field.

#### Parameters:

Parameter	Default value	Description
<code>time</code> *	—	Type: Object  Object with the set parameters.
<code>time.dayOfWeek</code>	—	Type: String  Abbreviated name of a day of the week. 'mon', 'tue', 'wed', 'thu', 'fri', 'sat', 'sun'.
<code>time.hours</code>	—	Type: Number  Hour.
<code>time.minutes</code>	—	Type: Number  Minutes.
<code>callback</code>	—	Type: Function  A function that is called after the time was set. Accepts a hash with the set data as input.

\* Mandatory parameter/option.

### Example:

```
// Creating a control that immediately shows the
// provider of "normal" traffic on the map.
var trafficControl = new ymaps.control.TrafficControl({
  shown: true,
  providerKey: 'traffic#archive'
});
map.controls.add(trafficControl);
// The local time will be set as soon as the provider
// gets data on the current time zone.
trafficControl.getProvider('traffic#archive').setTime({
  dayOfWeek: 'fri',
  hours: 9,
  minutes: 15
}, function (time) {
  alert('Local time ' + time.hours + ':' + time.minutes + ' set!');
});
```

### traffic.provider.Forecast

Extends [ITrafficProvider](#).

Provider for the traffic forecast. Accessible in the provider storage by the key 'traffic#forecast'.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
traffic.provider.Forecast([options[, state]])
```

Creates a provider for the traffic forecast.

### Parameters:

Parameter	Default value	Description
<a href="#">options</a>	—	Type: Object  Provider options. Options for provider layers are set via the options for the global collection of layers, <a href="#">Map.layers</a> . <ul style="list-style-type: none"><li>Options for the image layer <a href="#">Layer</a> are defined with the 'trafficImage' prefix.</li><li>Options for the hotspot layer <a href="#">hotspot.Layer</a> are defined with the 'trafficJam' prefix.</li></ul>
<a href="#">options.autoUpdate</a>	true	Type: Boolean  Flag that enables automatically updating traffic data. Automatic updates occur only when the "mousemove" event occurs every 4 minutes on the map. If this event does not occur, traffic stops being updated until there is a new event.
<a href="#">state</a>	—	Type: Object  Provider state.

Parameter	Default value	Description
<a href="#">state.timeOffset</a>	900	<p>Type: Number</p> <p>time interval between the current time and forecast time. Measured in seconds, in 15-minute intervals (900 seconds).</p> <pre>// Creating a traffic provider that // predicts traffic in half an hour, // and putting it on the map. var forecastProvider = new ymaps.traffic.provider.Forecast({}, {timeOffset: 30 * 60}); forecastProvider.setMap(myMap);  // Forbidding displaying popup // hints for traffic. myMap.layers.options.set({ // The option name is formed by // adding the 'trafficJam' prefix // to the hotspot layer option 'openHintOnHover':   trafficJamOpenHintOnHover:     false }); // ... // Removing the provider from the // map. forecastProvider.setMap(null);</pre>

## Fields

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	<p>Event manager.</p> <p>Inherited from <a href="#">IEventEmitter</a>.</p>
<a href="#">options</a>	<a href="#">IOptionManager</a>	<p>Options manager.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>
<a href="#">state</a>	<a href="#">data.Manager</a>	<p>Provider state. Names of fields that are available via the <a href="#">data.Manager.get</a> method:</p> <ul style="list-style-type: none"> <li><code>isInited</code> - Flag for whether the provider is ready to provide data.</li> <li><code>timestamp</code> - Current time in Unix timestamp format, in seconds.</li> <li><code>localtime</code> - Local time that the server is currently sending data for, in the format HH:MM.</li> <li><code>level</code> - Traffic level in points from 0 to 10.</li> <li><code>timeOffset</code> - Time interval between the current time and forecast time. Measured in seconds, in 15-minute intervals (900 seconds)</li> <li><code>isotime</code> - String containing the current date in the format "YYYY-MM-DDThh:mm:ss±hhmm".</li> </ul>

## Events

Name	Description
<a href="#">optionschange</a>	<p>Change to the object options.</p> <p>Inherited from <a href="#">ICustomizable</a>.</p>

**Methods**

Name	Returns	Description
<a href="#">getMap()</a>	<a href="#">Map</a>  null	Returns reference to the map.  Inherited from <a href="#">ITrafficProvider</a> .
<a href="#">getTime()</a>	Object null	Returns the time the user sees in the traffic control. The object has the following fields: <ul style="list-style-type: none"> <li>dayOfWeek - Abbreviations of days of the week. 'mon', 'tue', 'wed', 'thu', 'fri', 'sat', 'sun'.</li> <li>hours - Hours.</li> <li>minutes - Minutes.</li> </ul> <p>If the map center is located at a point that we cannot determine the time zone for, the function returns null; if we don't know which time zone we are in, we can't find out the local time.</p>
<a href="#">setMap(<a href="#">Reference</a>)</a>		

**Fields details****state**

```
{data.Manager} state
```

Provider state. Names of fields that are available via the `data.Manager.get` method:

- isInited - Flag for whether the provider is ready to provide data.
- timestamp - Current time in Unix timestamp format, in seconds.
- localtime - Local time that the server is currently sending data for, in the format HH:MM.
- level - Traffic level in points from 0 to 10.
- timeOffset - Time interval between the current time and forecast time. Measured in seconds, in 15-minute intervals (900 seconds)
- isotime - String containing the current date in the format "YYYY-MM-DDThh:mm:ss±hhmm".

**Methods details****getTime**

```
{Object|null} getTime()
```

**Returns** the time the user sees in the traffic control. The object has the following fields:

- dayOfWeek - Abbreviations of days of the week. 'mon', 'tue', 'wed', 'thu', 'fri', 'sat', 'sun'.
- hours - Hours.
- minutes - Minutes.

If the map center is located at a point that we cannot determine the time zone for, the function returns null; if we don't know which time zone we are in, we can't find out the local time.

**traffic.provider.storage**

Static object.

Instance of [util.Storage](#)

Storage for providers.

[Methods](#)

### Methods

Name	Returns	Description
<a href="#">add(key, object)</a>	<a href="#">util.Storage</a>	Adds an object to storage.
<a href="#">get(key)</a>	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
<a href="#">remove(key)</a>	<a href="#">util.Storage</a>	Deletes the "key: value" pair from storage.

## util

### util.AsyncStorage

Extends [util.Storage](#).

Storage that provides asynchronous access to key values.

[Constructor](#) | [Methods](#)

### Constructor

```
util.AsyncStorage()
```

### Methods

Name	Returns	Description
<a href="#">add(key, object)</a>	<a href="#">util.Storage</a>	Adds an object to storage. Inherited from <a href="#">util.Storage</a> .
<a href="#">define(key[, depends, resolveCallback[, context]])</a>	<a href="#">util.AsyncStorage</a>	Defines an asynchronous value in storage.
<a href="#">get(key)</a>	Object	Returns object stored for the specified key, or the primary key, if this is not a string. Inherited from <a href="#">util.Storage</a> .
<a href="#">isDefined(key)</a>	Boolean	Checking if the key can be accessed in the storage.
<a href="#">remove(key)</a>	<a href="#">util.Storage</a>	Deletes the "key: value" pair from storage. Inherited from <a href="#">util.Storage</a> .
<a href="#">require(keys[, successCallback[, errorCallback[, context]])</a>	<a href="#">vow.Promise</a>	Async request to get values from the storage.

## Methods details

### define

```
{util.AsyncStorage} define(key[, depends, resolveCallback[, context]])
```

Defines an asynchronous value in storage.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>key</code> *	—	Type: String  The key to use for making an async call.
<code>depends</code>	—	Type: String[]  Array of keys from the current storage that must be initialized before this key. This argument can be omitted.
<code>resolveCallback</code> *	—	Type: Function  Function that defines the value accessed by the key. The first argument in <code>resolveCallback</code> is the <code>provide</code> function to pass the value to. Invocation of the <code>provide</code> function can be deferred. The following arguments are values from storage that are specified in dependencies. The order of modules follows their order in the <code>depends</code> array.
<code>context</code>	—	Type: Object  Execution context for the function.

\* Mandatory parameter/option.

#### Examples:

##### 1.

```
asyncStorage
  .define('red', function (provide) {
    provide('#FF0000');
  });
```

##### 2.

```
asyncStorage
  .define('green', function (provide) {
    // The provide function can be called asynchronously.
    setTimeout(function () {
      provide('#008000');
    }, 400);
  });
```

## 3.

```

asyncStorage
  .define('yellow', function (provide) {
    provide('#FFFF00');
  })
  // To define the 'violet' key value, the 'yellow' key value is required.
  .define('violet', ['yellow'], function (provide, yellow) {
    console.log(yellow); // #FFFF00
    setTimeout(function () {
      provide('#9B30FF');
    }, 400);
  });

```

## 4.

```

var asyncStorage = new ymaps.util.AsyncStorage();
asyncStorage
  .define('red', function (provide) {
    provide('#FF0000');
  })
  .define('green', function (provide) {
    setTimeout(function () {
      provide('#008000');
    }, 400);
  })
  .define('yellow', function (provide) {
    provide('#FFFF00');
  })
  .define('violet', ['yellow'], function (provide, yellow) {
    setTimeout(function () {
      provide('#9B30FF');
    }, 400);
  });
// Requesting
asyncStorage.require(['red', 'green', 'violet'])
  .spread(function (red, green, violet) {
    console.log(red, green, violet); // #FF0000 #008000 #9B30FF
    // After the first async access, values can be accessed from the synchronous interface.
    console.log(asyncStorage.get('red'), asyncStorage.get('green'), asyncStorage.get('violet')); // #FF0000
#008000
// The value for the 'yellow' key is also in the storage now, because it was required in order to define
'violet'.
    console.log(asyncStorage.get('yellow')); // #FFFF00
  });

```

**isDefined**

```
{Boolean} isDefined(key)
```

Checking if the key can be accessed in the storage.

Returns true - defined, false - not.

**Parameters:**

Parameter	Default value	Description
<a href="#">key</a> *	—	Type: String  Key for the value.

\* Mandatory parameter/option.

**Example:**

```

if (asyncStorage.isDefined('red')) {
  asyncStorage.require('red')
    .spread(function (red) {
      // ...
    });
}

```

**require**

```
{vow.Promise} require(keys[, successCallback[, errorCallback[, context]])
```

Async request to get values from the storage.

**Returns** Promise that represents async access to the value.

**Parameters:**

Parameter	Default value	Description
<code>keys *</code>	—	Type: String String[]  Key or array of keys.
<code>successCallback</code>	—	Type: Function  Callback to invoke after getting all the values. Values from storage are passed to the function as arguments. The arguments are ordered according to their order in the keys array.
<code>errorCallback</code>	—	Type: Function  Callback to use if an error occurs. The error object is passed to the function.
<code>context</code>	—	Type: Object  Callback execution context.

\* Mandatory parameter/option.

**Examples:**

1.

```
asyncStorage.require(['green'])
  .spread(function (green) {
    // ...
  });
```

2.

```
var asyncStorage = new ymaps.util.AsyncStorage();
asyncStorage
  .define('red', function (provide) {
    provide('#FF0000');
  })
  .define('green', function (provide) {
    setTimeout(function () {
      provide('#008000');
    }, 400);
  });
// Requesting
asyncStorage.require(['red', 'green'])
  .spread(function (red, green) {
    console.log(red, green); // #FF0000 #008000
  });
```

## util.augment

Static function.

Base function implementing inheritance in JavaScript. Implements prototype inheritance without executing the parent constructor. The "superclass" field is appended to the child class, specifying the prototype of the parent class and the 'constructor' field, specifying the constructor of the class. Use the 'constructor' field of the 'superclass' object to refer to the constructor of the parent class.

**Returns** a prototype of the child class.

```
{ Object } util.augment(ChildClass, ParentClass, override)
```

#### Parameters:

Parameter	Default value	Description
<a href="#">ChildClass</a> *	—	Type: Function  Child class.
<a href="#">ParentClass</a> *	—	Type: Function  Parent class.
<a href="#">override</a> *	—	Type: Object  Set of additional fields and functions that will be appended to the prototype of the child class.

\* Mandatory parameter/option.

#### Example:

```
// Parent class
var ParentClass = function (param1, param2) {
  this.param1 = param1;
  this.param2 = param2;
};

ParentClass.prototype = {
  foo: function () {
    alert('Parent!');
  }
};

// Child class
var ChildClass = function (param1, param2, param3) {
  // Calling the parent's constructor
  ChildClass.superclass.constructor.call(this, param1, param2);
  this._param3 = param3;
};

// inheriting ChildClass from ParentClass
ymaps.util.augment(ChildClass, ParentClass, {
  // redefining the "foo" method in the descendant
  foo: function () {
    // Calling the parent class method
    ChildClass.superclass.foo.call(this);
    alert('Child!');
  }
});
```

## util.bind



**Attention:** This function is deprecated. Use native Function.bind method.

Static function.

Binds the passed function to the passed context.

**Returns** a copy of the given function with the specified `this` value.

```
{ Function } util.bind(callback, context)
```

#### Parameters:

Parameter	Default value	Description
<a href="#">callback</a> *	—	Type: Function  Function.

Parameter	Default value	Description
<code>context *</code>	—	Type: Object Execution context.

\* Mandatory parameter/option.

#### Example:

```
var myObject = {
  name: 'test!'
};
ymaps.geocode.load('Moscow')
  .then(ymaps.util.bind(function (res) {
    alert(this.name); // test!
  }, myObject));
```

## util.bounds

Static object.

Set of statistical methods for working with rectangular areas, represented as two opposite points in the coordinate system of the projection.

### Methods

#### Methods

Name	Returns	Description
<code>areIntersecting(bounds1, bounds2[, projection])</code>	Boolean	Determines whether two rectangular areas intersect.
<code>containsBounds(outer, inner[, projection])</code>	Boolean	Determines whether a rectangular area completely contains another rectangular area.
<code>containsPoint(bounds, point[, projection])</code>	Boolean	Determines whether a rectangular area contains a point.
<code>fromBounds(sourceBounds[, projection])</code>	Number[][]	Calculates the rectangular area that everything passed falls inside of.
<code>fromGlobalPixelBounds(pixelBounds, zoom[, projection])</code>	Number[][]	Converts a rectangular area from pixel coordinates to geo coordinates with the zoom factor taken into account.
<code>fromPoints(points[, projection])</code>	Number[][]	Calculates the minimal rectangular area that contains all the passed points.
<code>getCenter(bounds[, projection])</code>	Number[]	Calculates the center of a rectangular area in the coordinate system of the projection.
<code>getCenterAndZoom(bounds, containerSize[, projection, params])</code>	Object	Calculates the center and zoom that should be set for the map in order to display the passed area in its entirety.

Name	Returns	Description
<a href="#">getIntersections(bounds1, bounds2[, projection])</a>	Number[][]	Returns all intersections of two rectangular areas. If data is passed in geo coordinates, there may be more than one intersection. If the areas do not intersect, an empty array is returned.
<a href="#">getSize(bounds[, projection])</a>	Number[]	Calculates the dimensions of a rectangular area in the coordinate system of the projection.
<a href="#">toGlobalPixelBounds(geoBounds, zoom[, projection])</a>	Number[][]	Converts boundaries from geo coordinates to global pixels, accounting for the zoom level.

## Methods details

### areIntersecting

```
{Boolean} areIntersecting(bounds1, bounds2[, projection])
```

Determines whether two rectangular areas intersect.

**Returns** intersection attribute.

#### Parameters:

Parameter	Default value	Description
<a href="#">bounds1</a> *	—	Type: Number[][] First area.
<a href="#">bounds2</a> *	—	Type: Number[][] Second area.
<a href="#">projection</a>	<a href="#">projection.wgs84Mercator</a>	Type: <a href="#">IProjection</a> Projection.

\* Mandatory parameter/option.

### containsBounds

```
{Boolean} containsBounds(outer, inner[, projection])
```

Determines whether a rectangular area completely contains another rectangular area.

**Returns** inclusion attribute.

#### Parameters:

Parameter	Default value	Description
<a href="#">outer</a> *	—	Type: Number[][] External area

Parameter	Default value	Description
<code>inner *</code>	—	Type: <code>Number[][]</code> The area being checked.
<code>projection</code>	<code>projection.wgs84Mercator</code>	Type: <code>IProjection</code> Projection.

\* Mandatory parameter/option.

### containsPoint

```
{Boolean} containsPoint(bounds, point[, projection])
```

Determines whether a rectangular area contains a point.

**Returns** inclusion attribute.

#### Parameters:

Parameter	Default value	Description
<code>bounds *</code>	—	Type: <code>Number[][]</code> External area
<code>point *</code>	—	Type: <code>Number[]</code> The point being checked.
<code>projection</code>	<code>projection.wgs84Mercator</code>	Type: <code>IProjection</code> Projection.

\* Mandatory parameter/option.

### fromBounds

```
{Number[][]} fromBounds(sourceBounds[, projection])
```

Calculates the rectangular area that everything passed falls inside of.

**Returns** the calculated area.

#### Parameters:

Parameter	Default value	Description
<code>sourceBounds *</code>	—	Type: <code>Number[][][]</code> Array of rectangular areas
<code>projection</code>	<code>projection.wgs84Mercator</code>	Type: <code>IProjection</code> Projection.

\* Mandatory parameter/option.

### fromGlobalPixelBounds

```
{Number[][]} fromGlobalPixelBounds(pixelBounds, zoom[, projection])
```

Converts a rectangular area from pixel coordinates to geo coordinates with the zoom factor taken into account.

**Returns** calculated boundaries in geo coordinates.

**Parameters:**

Parameter	Default value	Description
<code>pixelBounds</code> *	—	Type: <code>Number[][]</code>  Original boundaries.
<code>zoom</code> *	—	Type: <code>Number</code>  Scale.
<code>projection</code>	<code>projection.wgs84Mercator</code>	Type: <code>IProjection</code>  The projection that will be used for calculating geo coordinates.

\* Mandatory parameter/option.

**fromPoints**

```
{Number[][]} fromPoints(points[, projection])
```

Calculates the minimal rectangular area that contains all the passed points.

**Returns** the calculated area.

**Parameters:**

Parameter	Default value	Description
<code>points</code> *	—	Type: <code>Number[][]</code>  Array of points.
<code>projection</code>	<code>projection.wgs84Mercator</code>	Type: <code>IProjection</code>  Projection.

\* Mandatory parameter/option.

**getCenter**

```
{Number[]} getCenter(bounds[, projection])
```

Calculates the center of a rectangular area in the coordinate system of the projection.

**Returns** center point in the coordinate system of the input data.

**Parameters:**

Parameter	Default value	Description
<code>bounds</code> *	—	Type: <code>Number[][]</code>  Area.
<code>projection</code>	<code>projection.wgs84Mercator</code>	Type: <code>IProjection</code>  Projection.

\* Mandatory parameter/option.

## getCenterAndZoom

```
{Object} getCenterAndZoom(bounds, containerSize[, projection[, params]])
```

Calculates the center and zoom that should be set for the map in order to display the passed area in its entirety.

**Returns** object with the center (Number[]) and zoom (Number) fields.

### Parameters:

Parameter	Default value	Description
<a href="#">bounds</a> *	—	Type: Number[][]  An area set in geographical coordinates. The first point contains the minimum values for latitude and longitude, and the second point contains the maximum values.
<a href="#">containerSize</a> *	—	Type: Number[]  Size of the map container.
<a href="#">projection</a>	<a href="#">projection.wgs84Mercator</a>	Type: <a href="#">IProjection</a>  Projection.
<a href="#">params</a>	—	Type: Boolean Object  Parameters or value of the <a href="#">preciseZoom</a> option.
<a href="#">params.inscribe</a>	true	Type: Boolean  If true, fit the area into the map; if false, fit the map into the area.
<a href="#">params.margin</a>	0	Type: Number Number[]  Offset from the borders of the visible area of the map. If a single number is set, it is applied to each side. If two numbers are set, they are the horizontal and vertical margins, respectively. If an array of four numbers is set, they are the top, right, bottom, and left margins.
<a href="#">params.preciseZoom</a>	false	Type: Boolean  When the value is "false", the zoom level is rounded down.

\* Mandatory parameter/option.

## getIntersections

```
{Number[][][]} getIntersections(bounds1, bounds2[, projection])
```

**Returns** all intersections of two rectangular areas. If data is passed in geo coordinates, there may be more than one intersection. If the areas do not intersect, an empty array is returned.

### Parameters:

Parameter	Default value	Description
<code>bounds1</code> *	—	Type: <code>Number[][]</code> First area.
<code>bounds2</code> *	—	Type: <code>Number[][]</code> Second area.
<code>projection</code>	<code>projection.wgs84Mercator</code>	Type: <code>IProjection</code> Projection.

\* Mandatory parameter/option.

## getSize

```
{Number[]} getSize(bounds[, projection])
```

Calculates the dimensions of a rectangular area in the coordinate system of the projection.

**Returns** size of the area.

### Parameters:

Parameter	Default value	Description
<code>bounds</code> *	—	Type: <code>Number[][]</code> Area.
<code>projection</code>	<code>projection.wgs84Mercator</code>	Type: <code>IProjection</code> Projection.

\* Mandatory parameter/option.

## toGlobalPixelBounds

```
{Number[][]} toGlobalPixelBounds(geoBounds, zoom[, projection])
```

Converts boundaries from geo coordinates to global pixels, accounting for the zoom level.

**Returns** resulting pixel boundaries.

### Parameters:

Parameter	Default value	Description
<code>geoBounds</code> *	—	Type: <code>Number[][]</code> Original boundaries.

Parameter	Default value	Description
<a href="#">zoom</a> *	—	Type: Number Scale.
<a href="#">projection</a>	<a href="#">projection.wgs84Mercator</a>	Type: <a href="#">IProjection</a> The projection in the coordinate system that the geo coordinates are set in.

\* Mandatory parameter/option.

## util.cursor

### util.cursor.Accessor

Object providing access to the cursor added to the map.

[Constructor](#) | [Methods](#)

### Constructor

```
util.cursor.Accessor(key)
```

### Parameters:

Parameter	Default value	Description
<a href="#">key</a> *	—	Type: String Key that corresponds to the cursor in the cursor storage.

\* Mandatory parameter/option.

### Methods

Name	Returns	Description
<a href="#">getKey()</a>	String	Returns the current key for access to the cursor in the cursor storage.
<a href="#">remove()</a>		Removes the cursor from the map.
<a href="#">setKey()</a>		Sets a new key for accessing the cursor. For this key, there should be a corresponding cursor in the cursor storage. If this cursor is active, it is immediately changed.

### Methods details

#### getKey

```
{String} getKey()
```

**Returns** the current key for access to the cursor in the cursor storage.

**remove**

```
{ } remove ()
```

Removes the cursor from the map.

**setKey**

```
{ } setKey ()
```

Sets a new key for accessing the cursor. For this key, there should be a corresponding cursor in the cursor storage. If this cursor is active, it is immediately changed.

**Parameters:**

Parameter	Default value	Description
<a href="#">key</a> *	—	Type: String

\* Mandatory parameter/option.

**util.cursor.Manager**

Cursor manager.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

**Constructor**

```
util.cursor.Manager (element)
```

Manager for cursors over a DOM element. Uses direct assignment via `style.cursor`.

**Parameters:**

Parameter	Default value	Description
<a href="#">element</a> *	—	Type: HTMLElement  The DOM element that cursors are being set for.

\* Mandatory parameter/option.

**Example:**

```
// putting the "arrow" cursor over the map
var cursor = map.cursors.push('arrow');
setTimeout(function () {
  // setting a different key for the cursor after 5 seconds
  cursor.setKey('grabbing');
  setTimeout(function () {
    // removing this cursor from the map after 5 more seconds
    cursor.remove();
  }, 5000);
}, 5000);
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">event.Manager</a>	Event manager for the cursor manager.

**Events**

Name	Description
<a href="#">change</a>	Change to the cursor.

**Methods**

Name	Returns	Description
<a href="#">push(key)</a>	<a href="#">util.cursor.Accessor</a>	Sets a new cursor and writes it to the stack of cursors for the object.

**Fields details****events**

```
{event.Manager} events
```

Event manager for the cursor manager.

**Events details****change**

Change to the cursor.

**Methods details****push**

```
{util.cursor.Accessor} push(key)
```

Sets a new cursor and writes it to the stack of cursors for the object.

**Returns** object providing access to the cursor added to the map.

**Parameters:**

Parameter	Default value	Description
<a href="#">key</a> *	—	Type: String  Cursor. Acceptable values: <ul style="list-style-type: none"> <li>"arrow" - Arrow cursor.</li> <li>"crosshair" - Crosshair cursor.</li> <li>"grab" - Hand cursor.</li> <li>"grabbing" - Closed hand.</li> <li>"help" - Arrow cursor with a question mark.</li> <li>"zoom" - Magnifying glass.</li> <li>"move" - Cursor consisting of four directional arrows.</li> <li>"pointer" - Pointing finger.</li> <li>"inherit" - Inherit the cursor from the parent.</li> </ul>

\* Mandatory parameter/option.

## util.defineClass

Static function.

Base function implementing class declaration in the Yandex.Maps API. Use this function to declare a new class, specify a set of methods for it and inherit from another class.

The "superclass" field is appended to the child class, specifying the prototype of the parent class. Use the 'constructor' field of the 'superclass' object to refer to the constructor of the parent class.

**Returns** class.

```
{ Object } util.defineClass(constructor[, parentClass[, override]])
```

**Parameters:**

Parameter	Default value	Description
<code>constructor</code> *	—	Type: Function Class constructor.
<code>parentClass</code>	—	Type: Function Parent class to inherit from. This argument may be omitted.
<code>override</code>	—	Type: Object Set of additional fields and functions that will be appended to the class prototype. There can be several sources (the function can have any number of parameters), and data is copied from right to left (the last argument has highest priority during copying).

\* Mandatory parameter/option.

### Examples:

1.

```
// Declaring a class with methods.
var MyClass = ymaps.util.defineClass(function () {
  this.field = 'fieldValue';
}, {
  doSomethingAwesome: function () {
    return 'methodResult';
  },
  stop: function () {
    //...
  }
});
var object = new MyClass();
console.log(object.field); // 'fieldValue'
console.log(object.doSomethingAwesome()); // 'methodResult'
```

2.

```
// Creating a custom class for a button, inherited from the base class of the button control.
// Pressing the button will switch the type of map tiles.
var CustomControl = ymaps.util.defineClass(function () {
  // Defining a limited set of options without a possibility to change them from outside.
  var data = {
    data: { content: 'Change the map type' },
    options: {
      selectOnClick: false,
      maxWidth: 150
    }
  };
  CustomControl.superclass.constructor.call(this, data);
```

```

}, ymaps.control.Button, {
  // Setting a list of class methods.

  // Overriding the button enable and disable methods.
  enable: function () {
    // You must call base class methods
    // to avoid breaking the button logic.
    CustomControl.superclass.enable.call(this);
    // Enabling and disabling button behavior.
    this.events.add('click', this.switchType, this);
  },

  disable: function () {
    this.events.remove('click', this.switchType, this);
    CustomControl.superclass.disable.call(this);
  },

  // Implementing our own methods.
  switchType: function () {
    var map = this.getMap();
    if (map) {
      if (map.getType() == 'yandex#map') {
        this.setSatelliteMapType();
      } else {
        this.setSchemeMapType();
      }
    }
  },

  setSchemeMapType: function () {
    var map = this.getMap();
    if (map) {
      map.setType('yandex#map');
    }
  },

  setSatelliteMapType: function () {
    var map = this.getMap();
    if (map) {
      map.setType('yandex#satellite');
    }
  }
});
// Instantiating a new class and adding it to the map.
var typeSwitcherButton = new CustomControl();
// The map creation code was omitted in this example.
myMap.controls.add(typeSwitcherButton);
// Calling the method of the class instance.
typeSwitcherButton.setSatelliteMapType();

```

## util.Dragger

Extends [IEventEmitter](#).

A special tool that provides a mechanism for dragging page elements. When using it, note that the mousemove and mouseup events will not register in the Yandex.Maps API events system when working with the dragger.

[Constructor](#) | [Fields](#) | [Events](#) | [Methods](#)

### Constructor

```
util.Dragger([params])
```

### Parameters:

Parameter	Default value	Description
<a href="#">params</a>	—	Type: Object  Dragger parameters.
<a href="#">params.autoStartElement</a>	—	Type: HTMLElement IDomEventEmitter  DOM element or implementation of the IDomEventEmitter interface, which starts the dragger when clicked.

Parameter	Default value	Description
<a href="#">params.byRightButton</a>	false	Type: Boolean  The right mouse button is used for starting the dragger using the 'autoStartElement' parameter. Dragging with the right button won't work on devices without mouse support.
<a href="#">params.tremor</a>	3	Type: Number  The minimum distance in pixels from the starting point necessary for starting the dragger.

**Example:**

```
// Example of dragging a DOM element to the map.
// A placemark is created where the DOM element is released.
var myMap = new ymaps.Map('map', {center: [35.76, 37.67], zoom: 5});
// The DOM element must have the CSS property position: absolute.
var element = document.getElementById('someId');
var dragger = new ymaps.util.Dragger({
  autoStartElement: element
});
var draggerEventsGroup = dragger.events.group();

draggerEventsGroup
  .add('start', function (event) {
    var pos = event.get('position');
    positionElement(pos[0], pos[1]);
    console.log('start');
  })
  .add('move', function (event) {
    var pos = event.get('position');
    positionElement(pos[0], pos[1]);
    console.log('move');
  })
  .add('stop', function (event) {
    draggerEventsGroup.removeAll();
    element.parentElement.removeChild(element);
    // Getting geographical coordinates at the point where the dragger stopped.
    var placemarkPosition = myMap.options.get('projection').fromGlobalPixels(
      myMap.converter.pageToGlobal(event.get('position')),
      myMap.getZoom()
    );
    myMap.geoObjects.add(
      new ymaps.Placemark(placemarkPosition)
    );
    console.log('stop');
  });

function positionElement (x, y) {
  element.style.left = x + 'px';
  element.style.top = y + 'px';
}
```

**Fields**

Name	Type	Description
<a href="#">events</a>	<a href="#">IEventManager</a>	Event manager.  Inherited from <a href="#">IEventEmitter</a> .

**Events**

Name	Description
<a href="#">cancel</a>	Cancels the dragger. This event occurs if the dragger stopped without sending the util.Dragger.start event. For example, if the click stopped without moving the mouse. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:

Name	Description
<a href="#">move</a>	<p>Changed position. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>delta - The difference between the locations of the current and previous dragger events.</li> </ul>
<a href="#">start</a>	<p>The dragger starts working. This event does not occur at the time of pressing, but at the first change to the position after pressing. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> </ul>
<a href="#">stop</a>	<p>The dragger stops working. This event cannot occur without the <a href="#">util.Dragger.start</a> event. Instance of the <a href="#">Event</a> class. Names of fields that are available via the <a href="#">Event.get</a> method:</p> <ul style="list-style-type: none"> <li>position - Coordinates relative to the document. Array in the format [pageX, pageY].</li> <li>delta - The difference between the locations of the current and previous dragger events.</li> </ul>

## Methods

Name	Returns	Description
<a href="#">destroy()</a>		Stops the dragger and deletes listening to the "mousedown" event for the "autoStartElement" event.
<a href="#">isDragging()</a>	Boolean	Returns whether the dragger is working now.
<a href="#">start(event)</a>		<p>Launches the dragger. This method is automatically called on the "mousedown" event for <code>autoStartElement</code>, if set. This method can be used for on-demand initialization. For example, when getting a response from the server. For correct functioning during on-demand initialization, the passed event must support the "position" field in the "get" method.</p>
<a href="#">stop()</a>		Stops the dragger. This method can be used for stopping the dragger prematurely.

## Events details

### cancel

Cancel the dragger. This event occurs if the dragger stopped without sending the `util.Dragger.start` event. For example, if the click stopped without moving the mouse. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

## move

Changed position. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- position - Coordinates relative to the document. Array in the format [pageX, pageY].
- delta - The difference between the locations of the current and previous dragger events.

## start

The dragger starts working. This event does not occur at the time of pressing, but at the first change to the position after pressing. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- position - Coordinates relative to the document. Array in the format [pageX, pageY].

## stop

The dragger stops working. This event cannot occur without the `util.Dragger.start` event. Instance of the [Event](#) class. Names of fields that are available via the [Event.get](#) method:

- position - Coordinates relative to the document. Array in the format [pageX, pageY].
- delta - The difference between the locations of the current and previous dragger events.

## Methods details

### destroy

```
{ } destroy()
```

Stops the dragger and deletes listening to the "mousedown" event for the "autoStartElement" event.

### isDragging

```
{Boolean} isDragging()
```

**Returns** whether the dragger is working now.

### start

```
{ } start(event)
```

Launches the dragger. This method is automatically called on the "mousedown" event for `autoStartElement`, if set. This method can be used for on-demand initialization. For example, when getting a response from the server. For correct functioning during on-demand initialization, the passed event must support the "position" field in the "get" method.

#### Parameters:

Parameter	Default value	Description
<a href="#">event</a> *	—	Type: <a href="#">IDomEvent</a> The event being initialized.

\* Mandatory parameter/option.

### stop

```
{ } stop()
```

Stops the dragger. This method can be used for stopping the dragger prematurely.

## util.extend

Static function.

Function that copies properties from one or several JavaScript objects to another JavaScript object.

```
util.extend(target, ...source)
```

### Parameters:

Parameter	Default value	Description
<code>target</code> *	—	Type: Object  Target JavaScript object. It will be modified as a result of the function.
<code>source</code> *	—	Type: Object  Source JavaScript object. All of its properties will be copied. There can be several sources (the function can have any number of parameters), and data is copied from right to left (the last argument has highest priority during copying).

\* Mandatory parameter/option.

### Example:

```
var options = ymaps.util.extend({
  prop1: 'a',
  prop2: 'b'
}, {
  prop2: 'c',
  prop3: 'd'
}, {
  prop3: 'e'
});
// We get the result: {
//   prop1: 'a',
//   prop2: 'c',
//   prop3: 'e'
// }
```

## util.hd

Static object.

Allows working with HD screens on various devices.

### Methods

#### Methods

Name	Returns	Description
<code>getPixelRatio()</code>	Number	Returns ratio of virtual and physical pixels on the screen.
<code>selectRatio(hash)</code>	Number	Returns result of selection — either the pixel ratio, or 1 if, for example, a string was passed.

Name	Returns	Description
<a href="#">selectValue(hash)</a>	Object	Returns an object from the passed hash that is equal to or closest to the pixel ratio key. If something other than a hash was passed, such as a string or function, then it is returned.

## Methods details

### getPixelRatio

```
{Number} getPixelRatio()
```

**Returns** ratio of virtual and physical pixels on the screen.

### selectRatio

```
{Number} selectRatio(hash)
```

**Returns** result of selection — either the pixel ratio, or 1 if, for example, a string was passed.

#### Parameters:

Parameter	Default value	Description
<a href="#">hash</a> *	—	Type: <a href="#">IRatioMap</a>  Object of type <a href="#">IRatioMap</a> for screens with different pixel ratios.

\* Mandatory parameter/option.

### selectValue

```
{Object} selectValue(hash)
```

**Returns** an object from the passed hash that is equal to or closest to the pixel ratio key. If something other than a hash was passed, such as a string or function, then it is returned.

#### Parameters:

Parameter	Default value	Description
<a href="#">hash</a> *	—	Type: <code>Object <a href="#">IRatioMap</a></code>  Object of type <a href="#">IRatioMap</a> for screens with different pixel ratios.

\* Mandatory parameter/option.

#### Example:

```
// We need to add a picture to the balloon so it looks good on HD screens.
var balloonImages = {
  // Normal picture.
  "100x100.png",
  // Picture for Retina.
  "2": "200x200.png"
};
var img = ymaps.util.hd.selectValue(balloonImages);
// Adding the point to the map.
```

```
myMap.geoObjects.add(new ymaps.GeoObject({
  geometry: "Point",
  coordinates: [55, 37],
  properties: {
    balloonContent:
      '';
  }
});
```

## util.math

### util.math.areEqual

Static function.

Compares two points with an error allowance.

**Returns** result of comparison.

```
{ Boolean } util.math.areEqual(first, second[, diff])
```

#### Parameters:

Parameter	Default value	Description
<code>first</code> *	—	Type: Number[] Array for comparison.
<code>second</code> *	—	Type: Number[] Array to compare.
<code>diff</code>	—	Type: Number Precision of comparison.

\* Mandatory parameter/option.

### util.math.cycleRestrict

Static function.

Assigns a numeric value to the set range. The range of values is considered to be closed in a ring. If a value is outside one of the ends of the range, the extra is counted off around the circle from the other end.

**Returns** value can be limited.

```
{ Number } util.math.cycleRestrict(value, min, max)
```

#### Parameters:

Parameter	Default value	Description
<code>value</code> *	—	Type: Number Value can be limited.
<code>min</code> *	—	Type: Number Minimum limit.
<code>max</code> *	—	Type: Number Maximum limit.

\* Mandatory parameter/option.

**Example:**

```
// Returns 110
ymaps.util.math.cycleRestrict(-250, -180, 180);
// Returns 60
ymaps.util.math.cycleRestrict(-300, -180, 180);
// Returns -170
ymaps.util.math.cycleRestrict(190, -180, 180);
```

**util.math.restrict**

Static function.

Restricts an input numeric value to the set minimum and maximum limits.

**Returns** value can be limited.

```
{ Number } util.math.restrict(value, min, max)
```

**Parameters:**

Parameter	Default value	Description
<code>value</code> *	—	Type: Number Value can be limited.
<code>min</code> *	—	Type: Number Minimum limit.
<code>max</code> *	—	Type: Number Maximum limit.

\* Mandatory parameter/option.

**Example:**

```
ymaps.util.math.restrict(-250, -180, 180) = -180
```

**util.pixelBounds**

Static object.

**Methods****Methods**

Name	Returns	Description
<code>areIntersecting(bounds1, bounds2)</code>	Boolean	Determines whether two rectangular areas intersect.
<code>containsBounds(outer, inner)</code>	Boolean	Determines whether a rectangular area completely contains another rectangular area.
<code>containsPoint(bounds, point)</code>	Boolean	Determines whether a rectangular area contains a point.

Name	Returns	Description
<a href="#">fromBounds(sourceBounds)</a>	Number[][]	Calculates the rectangular area that everything passed falls inside of.
<a href="#">fromPoints(points)</a>	Number[][]	Calculates the rectangular area that the passed points fall inside of.
<a href="#">getCenter(bounds)</a>	Number[]	Calculates the center of the rectangular area.
<a href="#">getIntersection(bounds1, bounds2)</a>	Number[][] null	Calculates the intersection of two rectangular areas.
<a href="#">getSize(bounds)</a>	Number[]	Calculates the dimensions of a rectangular area.

## Methods details

### areIntersecting

```
{Boolean} areIntersecting(bounds1, bounds2)
```

Determines whether two rectangular areas intersect.

**Returns** intersection attribute.

#### Parameters:

Parameter	Default value	Description
<a href="#">bounds1</a> *	—	Type: Number[][] First area.
<a href="#">bounds2</a> *	—	Type: Number[][] Second area.

\* Mandatory parameter/option.

### containsBounds

```
{Boolean} containsBounds(outer, inner)
```

Determines whether a rectangular area completely contains another rectangular area.

**Returns** inclusion attribute.

#### Parameters:

Parameter	Default value	Description
<a href="#">outer</a> *	—	Type: Number[][] External area

Parameter	Default value	Description
<code>inner</code> *	—	Type: Number[][] The area being checked.

\* Mandatory parameter/option.

### containsPoint

```
{Boolean} containsPoint(bounds, point)
```

Determines whether a rectangular area contains a point.

**Returns** inclusion attribute.

#### Parameters:

Parameter	Default value	Description
<code>bounds</code> *	—	Type: Number[][] External area
<code>point</code> *	—	Type: Number[] The point being checked.

\* Mandatory parameter/option.

### fromBounds

```
{Number[][]} fromBounds(sourceBounds)
```

Calculates the rectangular area that everything passed falls inside of.

**Returns** the calculated area.

#### Parameters:

Parameter	Default value	Description
<code>sourceBounds</code> *	—	Type: Number[][][] Array of rectangular areas

\* Mandatory parameter/option.

### fromPoints

```
{Number[][]} fromPoints(points)
```

Calculates the rectangular area that the passed points fall inside of.

**Returns** the calculated area.

#### Parameters:

Parameter	Default value	Description
<code>points</code> *	—	Type: <code>Number[][]</code> Array of points.

\* Mandatory parameter/option.

### getCenter

```
{Number[]} getCenter(bounds)
```

Calculates the center of the rectangular area.

**Returns** center point in the coordinate system of the input data.

#### Parameters:

Parameter	Default value	Description
<code>bounds</code> *	—	Type: <code>Number[][]</code> Area.

\* Mandatory parameter/option.

### getIntersection

```
{Number[][]|null} getIntersection(bounds1, bounds2)
```

Calculates the intersection of two rectangular areas.

**Returns** the rectangular area that is formed by the intersection of the passed areas, or `null` if the areas do not intersect.

#### Parameters:

Parameter	Default value	Description
<code>bounds1</code> *	—	Type: <code>Number[][]</code> First area.
<code>bounds2</code> *	—	Type: <code>Number[][]</code> Second area.

\* Mandatory parameter/option.

### getSize

```
{Number[]} getSize(bounds)
```

Calculates the dimensions of a rectangular area.

**Returns** size of the area.

#### Parameters:

Parameter	Default value	Description
<code>bounds</code> *	—	Type: <code>Number[][]</code> Area.

\* Mandatory parameter/option.

## `util.requireCenterAndZoom`

Static function.

Calculates the optimal center and zoom level of the map to display the passed area on the specified type of map. The zoom level will be within the available zoom range.

**Returns** the promise object, which will be confirmed by an object with the center and zoom fields, or will be rejected with an error.

```
{ vow.Promise } util.requireCenterAndZoom(mapType, bounds, containerSize[, params])
```

### Parameters:

Parameter	Default value	Description
<code>mapType</code> *	—	Type: <code>String MapType map.ZoomRange</code>  Map type. Key string from <code>mapType.storage</code> , or an instance of the <code>MapType</code> class. Or the manager of map zoom coefficients for a specific map.
<code>bounds</code> *	—	Type: <code>Number[][]</code>  An area set in geographical coordinates. The first point contains the minimum values for latitude and longitude, and the second point contains the maximum values.
<code>containerSize</code> *	—	Type: <code>Number[]</code>  Size of the map container.
<code>params</code>	—	Type: <code>Object</code>  Parameters.
<code>params.inscribe</code>	<code>true</code>	Type: <code>Boolean</code>  If <code>true</code> , fit the area into the map; if <code>false</code> , fit the map into the area.
<code>params.margin</code>	<code>0</code>	Type: <code>Number Number[]</code>  Offset from the borders of the visible area of the map. If a single number is set, it is applied to each side. If two numbers are set, they are the horizontal and vertical margins, respectively. If an array of four numbers is set, they are the top, right, bottom, and left margins.

Parameter	Default value	Description
<a href="#">params.preciseZoom</a>	false	Type: Boolean  When the value is "false", the zoom level is rounded down.

\* Mandatory parameter/option.

#### Example:

```
// Finding the optimal center and zoom level of the map..
ymaps.util.requireCenterAndZoom(
  myMap.getType(),
  [[50.531219,31.278264], [50.966841,31.964909]],
  myMap.container.getSize()
).then(function (result) {
  // Setting the optimal center and zoom level of the map.
  myMap.setCenter(result.center, result.zoom);
});
```

## util.Storage

Object storage by keys.

[Constructor](#) | [Methods](#)

### Constructor

```
util.Storage()
```

### Methods

Name	Returns	Description
<a href="#">add(key, object)</a>	<a href="#">util.Storage</a>	Adds an object to storage.
<a href="#">get(key)</a>	Object	Returns object stored for the specified key, or the primary key, if this is not a string.
<a href="#">remove(key)</a>	<a href="#">util.Storage</a>	Deletes the "key: value" pair from storage.

### Methods details

#### add

```
{util.Storage} add(key, object)
```

Adds an object to storage.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<a href="#">key</a> *	—	Type: String  Key.
<a href="#">object</a> *	—	Type: Object  Stored object.

\* Mandatory parameter/option.

### get

```
{Object} get (key)
```

**Returns** object stored for the specified key, or the primary key, if this is not a string.

#### Parameters:

Parameter	Default value	Description
<code>key</code> *	—	Type: String Object Key.

\* Mandatory parameter/option.

### remove

```
{util.Storage} remove (key)
```

Deletes the "key: value" pair from storage.

**Returns** self-reference.

#### Parameters:

Parameter	Default value	Description
<code>key</code> *	—	Type: String Key.

\* Mandatory parameter/option.

## VOW

Static object.

Contains methods for creating and processing promise objects.

#### Note:

This class is a part of the [Vow](#) library.

Only some of the methods are described below. The complete list of methods is available here: <http://dfilatov.github.io/vow/>.

Copyright (c) 2012-2013 Filatov Dmitry ([dfilatov@yandex-team.ru](mailto:dfilatov@yandex-team.ru)). Dual licensed under the [MIT](#) and [GPL](#) licenses.

#### Methods

#### Methods

Name	Returns	Description
<code>all(iterable)</code>	<a href="#">vow.Promise</a>	Returns a promise object that will be either allowed or denied only if all the specified objects will be accepted or rejected, correspondingly.

Name	Returns	Description
<code>defer()</code>	<a href="#">vow.Deferred</a>	Creates a new deferred object. The same as <code>`new ymaps.vow.Deferred()`</code> .
<code>reject(reason)</code>	<a href="#">vow.Promise</a>	Returns a promise object rejected with the specified reason.
<code>resolve(value)</code>	<a href="#">vow.Promise</a>	Returns the promise object which is accepted with the specified value.

## Methods details

### all

```
{vow.Promise} all(iterable)
```

**Returns** a promise object that will be either allowed or denied only if all the specified objects will be accepted or rejected, correspondingly.

#### Parameters:

Parameter	Default value	Description
<code>iterable *</code>	—	Type: <code>Object Object[]</code>  Set of promise objects and/or values.

\* Mandatory parameter/option.

#### Examples:

##### 1.

```
var deferred1 = ymaps.vow.defer();
var deferred2 = ymaps.vow.defer();

ymaps.vow.all([deferred1.promise(), deferred2.promise(), 3])
  .then(function(value) {
    // value => [1, 2, 3]
  });

deferred1.resolve(1);
deferred2.resolve(2);
```

##### 2.

```
var deferred1 = ymaps.vow.defer();
var deferred2 = ymaps.vow.defer();

ymaps.vow.all({ p1 : deferred1.promise(), p2 : deferred2.promise(), p3 : 3 })
  .then(function(value) {
    // value => { p1 : 1, p2 : 2, p3 : 3 }
  });

deferred1.resolve(1);
deferred2.resolve(2);
```

### defer

```
{vow.Deferred} defer()
```

Creates a new deferred object. The same as ``new ymaps.vow.Deferred()``.

See [vow.Deferred](#)

Returns a deferred object.

### reject

```
{vow.Promise} reject(reason)
```

Returns a promise object rejected with the specified reason.

#### Parameters:

Parameter	Default value	Description
<code>reason</code> *	—	Type: Object The reason for rejection.

\* Mandatory parameter/option.

### resolve

```
{vow.Promise} resolve(value)
```

Returns the promise object which is accepted with the specified value.

#### Parameters:

Parameter	Default value	Description
<code>value</code> *	—	Type: Object Value.

\* Mandatory parameter/option.

## vow.Deferred

A class which describes the deferred objects.

**Note:** This class is a part of the [Vow](http://dfilatov.github.io/vow/) library. Only some of the methods are described below. The complete list of methods is available here: <http://dfilatov.github.io/vow/>. Copyright (c) 2012-2013 Filatov Dmitry ([dfilatov@yandex-team.ru](mailto:dfilatov@yandex-team.ru)). Dual licensed under the [MIT](#) and [GPL](#) licenses.

**Note:** It is not a stand-alone module: it is available only if the [vow](#) module is connected.

[Constructor](#) | [Methods](#)

### Constructor

```
vow.Deferred()
```

Creates a deferred object.

#### Example:

```
function someAsyncMethod () {
  var deferred = new ymaps.vow.Deferred(); // or `var deferred = ymaps.vow.defer();`

  doSomeAsyncStuff(function (err, value) {
    if (err) {
      deferred.reject(err);
      return;
    }

    deferred.resolve(value);
  });

  return deferred.promise();
}
```

```
someAsyncMethod().then(function (value) {
  console.log('The method result: ' + value);
}, function (err) {
  console.log('Error: ' + err);
});
```

## Methods

Name	Returns	Description
<code>promise()</code>	<code>vow.Promise</code>	Returns the associated promise object.
<code>reject(reason)</code>		Rejects the associated promise object with the specified reason.
<code>resolve(value)</code>		Accepts the associated promise object with the specified value.

## Methods details

### promise

```
{vow.Promise} promise()
```

Returns the associated promise object.

### reject

```
{ } reject(reason)
```

Rejects the associated promise object with the specified reason.

#### Parameters:

Parameter	Default value	Description
<code>reason *</code>	—	Type: Object The reason for rejection.

\* Mandatory parameter/option.

### resolve

```
{ } resolve(value)
```

Accepts the associated promise object with the specified value.

#### Parameters:

Parameter	Default value	Description
<code>value *</code>	—	Type: Object Value.

\* Mandatory parameter/option.

## vow.Promise

A class which describes the promise objects.

[The Promise/A+ specification.](#)

**Note:** This class is a part of the [Vow](#) library. Only some of the methods are described below. The complete list of methods is available here: <http://dfilatov.github.io/vow/>. Copyright (c) 2012-2013 Filatov Dmitry ([dfilatov@yandex-team.ru](mailto:dfilatov@yandex-team.ru)). Dual licensed under the [MIT](#) and [GPL](#) licenses.

**Note:** It is not a stand-alone module: it is available only if the [vow](#) module is connected.

[Constructor](#) | [Methods](#)

### Constructor

```
vow.Promise([resolver])
```

Creates a promise object.

#### Parameters:

Parameter	Default value	Description
<a href="#">resolver</a>	—	Type: Function  The function which takes the resolve and reject methods as parameters to set status and values for the created promise object.

#### Example:

```
function someAsyncMethod () {
  return new ymaps.vow.Promise(function (resolve, reject) {
    doSomeAsyncStuff(function (err, value) {
      if (err) {
        reject(err);
        return;
      }

      resolve(value);
    });
  });
}

someAsyncMethod().then(function (value) {
  console.log('The method result: ' + value);
}, function (err) {
  console.log('Error: ' + err);
});
```

### Methods

Name	Returns	Description
<a href="#">done</a> ([onFulfilled[, onRejected[, onProgress[, ctx]]]])		Similar to the method <a href="#">vow.Promise.then</a> , which closes the promise chain. Throws an exception if the promise object is rejected.

Name	Returns	Description
<code>spread([onFulfilled[, onRejected[, ctx]])</code>	<code>vow.Promise</code>	Similar to the method <code>vow.Promise.then</code> , which calls the callback functions with a set of arguments (corresponding to an array) for which the promise object can be either allowed or rejected. Usually it is used in combination with methods like <code>vow.all</code> .
<code>then([onFulfilled[, onRejected[, onProgress[, ctx]])</code>	<code>vow.Promise</code>	Specifies a handler function for a promise object.
<code>valueOf()</code>	Object	Returns a value for an allowed promise object or a rejection reason for a rejected one.

## Methods details

### done

```
{ } done([onFulfilled[, onRejected[, onProgress[, ctx]])
```

Similar to the method `vow.Promise.then`, which closes the promise chain. Throws an exception if the promise object is rejected.

#### Parameters:

Parameter	Default value	Description
<code>onFulfilled</code>	—	Type: Function  The callback function which will be called if the promise object is resolved.
<code>onRejected</code>	—	Type: Function  The callback function which will be called if the promise object is rejected.
<code>onProgress</code>	—	Type: Function  The callback function which will be called when "notifying" the promise object.
<code>ctx</code>	—	Type: Object  The execution context of the callback functions.

#### Example:

```
var deferred = ymaps.vow.defer();
deferred.reject(Error('Internal error'));
deferred.promise().done(); // Throws an exception.
```

## spread

```
{vow.Promise} spread([onFulfilled[, onRejected[, ctx]])
```

Similar to the method [vow.Promise.then](#), which calls the callback functions with a set of arguments (corresponding to an array) for which the promise object can be either allowed or rejected. Usually it is used in combination with methods like [vow.all](#).

**Returns** a new promise object.

### Parameters:

Parameter	Default value	Description
<a href="#">onFulfilled</a>	—	Type: Function  The callback function which will be called if the promise object is resolved.
<a href="#">onRejected</a>	—	Type: Function  The callback function which will be called if the promise object is rejected.
<a href="#">ctx</a>	—	Type: Object  The execution context of the callback functions.

### Example:

```
var deferred1 = ymaps.vow.defer();
var deferred2 = ymaps.vow.defer();

ymaps.vow.all([deferred1.promise(), deferred2.promise()]).spread(function(arg1, arg2) {
  // arg1 => 1, arg2 => 'two'
});

deferred1.resolve(1);
deferred2.resolve('two');
```

## then

```
{vow.Promise} then([onFulfilled[, onRejected[, onProgress[, ctx]])
```

Specifies a handler function for a promise object.

**Returns** a new promise object. See the [specification](#).

### Parameters:

Parameter	Default value	Description
<a href="#">onFulfilled</a>	—	Type: Function  The callback function which will be called if the promise object is resolved.
<a href="#">onRejected</a>	—	Type: Function  The callback function which will be called if the promise object is rejected.

Parameter	Default value	Description
<code>onProgress</code>	—	Type: Function  The callback function which will be called when "notifying" the promise object.
<code>ctx</code>	—	Type: Object  The execution context of the callback functions.

### valueOf

```
{Object} valueOf()
```

**Returns** a value for an allowed promise object or a rejection reason for a rejected one.