

Maps API for JavaScript

**Enterprise Version 2.5.4
Developer's Guide**

Contents

Legal Notices.....	4
Document Information.....	5
 Chapter 1: Overview.....	 6
What Is the Maps API for JavaScript?.....	7
Feature List.....	7
Packages and Detection.....	8
Browser Support.....	9
HTML5 Support.....	9
Performance.....	10
 Chapter 2: Quick Start.....	 11
Basic Scenario.....	12
Loading the API Code Library.....	12
Customer Integration Testing.....	13
URLs to Use.....	14
Implementing the Application.....	14
Setting Credentials.....	15
Displaying a Map.....	15
 Chapter 3: User Guide.....	 17
Acquiring Credentials.....	18
Beyond the Basic Map Application.....	18
Map Types.....	20
Components and User Interaction.....	23
Markers.....	25
Geo Objects.....	28
Map Overlays.....	29

here

A Map of Berlin from 1789.....	30
Routing.....	33
Traffic.....	37
KML Support.....	38
Heat Maps.....	39
Marker Clustering.....	41
Positioning.....	43
Event System.....	45
Observers and Mutable Objects.....	49
Multi-language Support.....	50
 Chapter 4: API reference.....	 52
nokia.....	53
Features.....	53
Settings.....	58
maps.....	62

here

Legal Notices

© 2014 HERE. All rights reserved.

This material, including documentation and any related computer programs, is protected by copyright controlled by HERE. All rights are reserved. Copying, including reproducing, storing, adapting or translating, any or all of this material requires the prior written consent of HERE. This material also contains confidential information, which may not be disclosed to others without the prior written consent of HERE.

Trademark Acknowledgements

HERE and Nokia are trademarks or registered trademarks of Nokia Corporation in the United States and other countries.

Other trade names are trademarks or registered trademarks of their owners.

Disclaimer

This content is provided "as-is" and without warranties of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability, fitness for a particular purpose, satisfactory quality and non-infringement. Nokia does not warrant that the content is error free and Nokia does not warrant or make any representations regarding the quality, correctness, accuracy, or reliability of the content. You should therefore verify any information contained in the content before acting on it.

To the furthest extent permitted by law, under no circumstances, including without limitation Nokia's negligence, shall Nokia be liable for any damages, including, without limitation, direct, special, indirect, punitive, consequential, exemplary and/ or incidental damages that result from the use or application of this content, even if Nokia or an authorized representative has been advised of the possibility of such damages.

here

Document Information

Product

Name: Maps API for JavaScript

Version: Enterprise Version 2.5.4

Document

Name: Maps API for JavaScript Developer's Guide

Id: e2cf024-1391695836

Status: FINAL

Date: 2014-Feb-06, 14:15 (GMT)

here

Chapter 1

Overview

Topics:

- [What Is the Maps API for JavaScript?](#)
- [Feature List](#)
- [Packages and Detection](#)
- [Browser Support](#)
- [HTML5 Support](#)
- [Performance](#)

This article defines the HERE Maps API for JavaScript, explains its purpose, summarizes its features and outlines how it supports third-party application development.

What Is the Maps API for JavaScript?

The HERE Maps API for JavaScript (the *Maps API*) is a set of programming interfaces that enable developers to build Web applications with feature rich, interactive [HERE Maps](#) at their center. The API consists of libraries of classes and methods with which to implement the functionality of an interactive application.

The associated [API Explorer](#) offers ready-made working examples, whose code the developer can modify and view the effect immediately.

Feature List

The main features offered by the Maps API are summarized in the table below:

Table 1: Main features of the HERE Maps API for JavaScript

Feature	Description
HERE Maps map data	The API provides full access to world-leading map data in three different modes: <i>satellite</i> , <i>terrain</i> and <i>hybrid</i> .
Points of interest	The map data to which the API offers access, include millions of different places, landmarks, places and addresses located on maps. Their visibility on the map can be controlled via the API.
Searches	The API allows you to build search functionality into you application. Users can searches for places, using keywords, addresses, or geographic coordinates.
Geocoding	The API provides full access to geocoding and reverse geocoding services.
Routing	The API allows you to define and render routes between a start and a destination point. It supports many navigation options such as toll road preference and transport type.
W3C positioning	The API includes built-in functionality that takes advantage of the W3C Geolocation API supported by many browsers.
Custom items	The API allows you to modify existing markers or create custom ones, using SVG or bitmap images. You can also add geo shapes based on coordinates to the map: polygons, polylines, circles or rectangles, and make these custom map object interactive by assigning UI events to them.

Packages and Detection

The HERE Maps API for JavaScript is modular and contains separate packages for maps, places, positioning, directions (routing) and data rendering. If an application does not require certain packages, you can reduce the footprint and load time by excluding them.

The entry point to the API is the JavaScript file "jsl.js" (see also [Quick Start](#) on page 11). It contains a package loader and an environment detection mechanism. To load the API, set the "src" attribute of a <script> element to the base URL followed by the name of this file. You can alter the packages loaded by the <script> element by appending the parameter with, followed by the equals sign (=), followed by a comma-separated list of packages. If you do not specify any packages (that is if you use the URL only without the with parameter), the Maps library package is loaded by default.

The table below shows the packages you can select, using the query parameter with and the default loading behavior.

Table 2: Selectable packages in the HEREMaps API for JavaScript

URL Parameter	Allowed Values [1]	Default
with	maps, positioning, directions, datarendering, all [2]	maps

The following notes relate to this table:

- [1] values can be combined in a comma-separated sequence; all must not be used if any other package is specified
- [2] all loads all the available libraries

The example below shows an HTML page that loads the following library packages: Maps, Places (complete), Routing and Positioning.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html;
    charset=UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=7;
    IE=EmulateIE9; IE=10" />
    <script type="text/javascript"
      src="http://js.api.here.com/ee/2.5.4/jsl.js?

```



```

        with=maps,positioning">
    </script>
</head>
<body>
    ...
</body>
</html>

```

Please note that the last part of the URL before the name of the JavaScript file is the release version number for the Maps API, which means that the URL changes, depending on which version you wish to use.

Browser Support

The Maps API is a multi-purpose and cross-platform API, targeting all JavaScript-enabled Web browsers, whether desktop or mobile, as well as all HTML5 environments (see the next section for information about HTML5 support).

Because the number of possible browser and platform combinations is too large, we limit our support to the following list of browsers:

Table 3: Browser support in HERE Maps API for JavaScript

Browser	Support
Internet Explorer 7+	desktop, Windows Phone with Mango update or later on mobile*
Firefox 3.6+	desktop
Google Chrome 12+	desktop, Android 2.2+ on mobile/tablet*
Apple Safari 5+	desktop, iOS 4+ on mobile/tablet

* support does not extend to all multi-touch gestures

HTML5 Support

Although HTML5 is a new emerging HTML standard, and as such it is still undergoing continual improvement, the HERE Maps API for JavaScript is committed to supporting it.

For information regarding the latest features of HTML5 please visit the official [HTML5 Specification page of the W3C](#). Details on HTML5 compliance in different Web browsers, are available on [THE HTML5 TEST](#).

here

The built-in support for HTML5 in the Maps API enhances the user experience in browsers that implement the following aspects of the HTML5 specification:

SVG	The HTML5 specification allows for embedding of Scalable Vector Graphics (SVG) in HTML pages. The use of SVG increases the rendering speed of custom shapes and forms. The Maps API supports the use of SVG icons as custom markers.
Geolocation	Browsers that offer HTML5 geolocation support are able to share their position on the globe and this means that the Maps API can request the location details from the browser. On most mobile devices, geolocation detects a user's position to within a few meters. It is also possible to obtain an accurate current position on desktop environments connected to the Internet via Wi-Fi, but note that there may be some delay in response, as the Wi-Fi positioning look-up is based on a server request. Positioning look-up and response time vary from browser to browser. The security policy on most browsers does not automatically allow immediate access and a confirmation dialog is shown instead. Please bear this in mind when using the geolocation feature.

Performance

The HERE Maps API for JavaScript has been designed to meet the demands of modern Web application development. The common development patterns it implements help improve efficiency and shorten the development cycle.

Fast start-up	The Maps API for JavaScript is outstanding in terms of memory consumption due to the small size of the underlying JavaScript framework. The map starts fast and the content loads smoothly.
Fast loading	The Maps API for JavaScript provides packaged and lazy loading of features such as route and search – and even of the map data itself. The content is downloaded from the server only when needed.
Fast browsing	The Maps API for JavaScript supports in-built browser caching. Once the content has been downloaded from the server, it is reused as much as possible to minimize bandwidth usage.

The HERE logo, consisting of the word "here" in a blue, lowercase, sans-serif font, tilted at an angle.

Chapter 2

Quick Start

Topics:

- [Basic Scenario](#)
- [Loading the API Code Library](#)
- [Customer Integration Testing](#)
- [Implementing the Application](#)

This article describes the simplest scenario using the Maps API for JavaScript and illustrates it with a working example.

Note that the example uses demo credentials. These are suitable for testing, but must be replaced with application-specific credentials in your own applications.(please see also [Acquiring Credentials](#) on page 18)

here

Basic Scenario

The most basic scenario is to create an application that displays a non-interactive map. To do that:

1. Load the Maps API for JavaScript library from the HTML page – the "src" attribute of a `<script>` element within the `<head>` must point to the Maps package.
2. Set authentication and authorization credentials – this is needed for unlimited access to the HERE Maps API for JavaScript.
3. Implement the application – write the code that implements the behavior of the application inside a `<script>` element in the `<body>` of the page; it takes only a few lines of JavaScript to create a fully functioning application with an interactive map.

Loading the API Code Library

To load the Maps library of the HERE Maps API for JavaScript, add the following `<script>` element to the `<head>` of the HTML document:

```
<script src="http://js.api.here.com/ee/2.5.4/js1.js" type="text/
javascript" charset="utf-8"></script>
```

The URL in the "src" attribute contains the version number, which is specific to the API release. In other words, that part of the URL changes with each release of the Maps API.

If you use Microsoft Internet Explorer, you can ensure compatibility by placing this line with a meta tag before any `<script>` elements in the `<head>`:

```
<meta http-equiv="X-UA-Compatible" content="IE=7; IE=EmulateIE9; IE=10"/>
```

And here is the complete `<head>` element that loads the default maps library package of HERE Maps API for JavaScript and makes sure there is no conflict with IE:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    ...
    <meta http-equiv="X-UA-Compatible" content="IE=7;
      IE=EmulateIE9; IE=10" />
    <script src="http://js.api.here.com/ee/2.5.4/js1.js"
      type="text/javascript" charset="utf-8">
    </script>
    ...
  </head>
</html>
```

here

```
</head>
```

The file `jsl.js` contains:

- common classes that are always needed, regardless of the module combination
- a package loader which enables the loading of specific module combinations
- environment detection code that enables the API to determine an optimal set of packages for the environment

For more information about loading the library and the available options, please see [Packages and Detection](#) on page 8.

Note that if you wish the map to fill the entire window in Internet Explorer 7 or 8, you must set the CSS attributes that control the position and dimensions of the body element as shown in the example below, otherwise map will flicker during panning.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
...
  <meta http-equiv="X-UA-Compatible" content="IE=7; IE=EmulateIE9; IE=10" />
  <style type="text/css">
    html {
      overflow:hidden;
    }

    body {
      margin: 0;
      padding: 0;
      position: absolute;
      overflow:hidden;
      width: 100%;
      height: 100%;
    }

    #mapContainer {
      width: 100%;
      height: 100%;
      left: 0;
      top: 0;
      position: absolute;
    }
  </style>
...
</head>
<body>
  <div id="mapContainer"></div>
...
</body>
```

Customer Integration Testing

HERE is committed to maintain the best possible production service for all customers. Given that the production environment is live and common to all API users, we request that you use the alternative

here

Customer Integration Testing (CIT) environment when evaluating our products, running tests, making changes in your code and altering the way you access our APIs.

The CIT environment also allows you to test your software against a newer version of the service before HERE brings that version into production. CIT offers a fully functional environment for customers to use for development and testing, but it does not support high loads or performance testing in general.

Note that the same application id can be used in both environments, but CIT may require a dedicated application code. If this is the case, please contact us as described under [Service Support](#).

The CIT environment is not intended for general production use.

URLs to Use

For the purposes of testing and evaluation, include the following line in the <head> element of the HTML page to load the API.

```
<script src="http://js.cit.api.here.com/ee/2.5.4/js1.js" type="text/
javascript" charset="utf-8"></script>
```

In addition, place the following line, which selects the staging mode, in the <script> element within the <body> of the HTML page (see also [Implementing the Application](#) on page 14) before you initiate the map:

```
nokia.Settings.set("serviceMode", "cit");
```

To use the production environment, include this line.

```
<script src="http://js.api.here.com/ee/2.5.4/js1.js" type="text/
javascript" charset="utf-8"></script>
```

Implementing the Application

At this point, two steps from the [Basic Scenario](#) on page 12 remain to be completed to ensure that the application is complete and functional: we need to provide access credentials and the code that defines what the application does. To do this, we need to place a <script> element within the <body> of the HTML page and write the implementation code there.

here

Setting Credentials

An essential part of creating a working application with the HERE Maps API for JavaScript is to provide the application credentials. To do that, use the method `nokia.Settings()` – note that you need to use the actual `app_id` and `app_code` you have received on registration (please see [Acquiring Credentials](#) on page 18):

```
nokia.Settings.set("app_id", "YOUR_APPID");  
nokia.Settings.set("app_code", "YOUR_TOKEN");
```

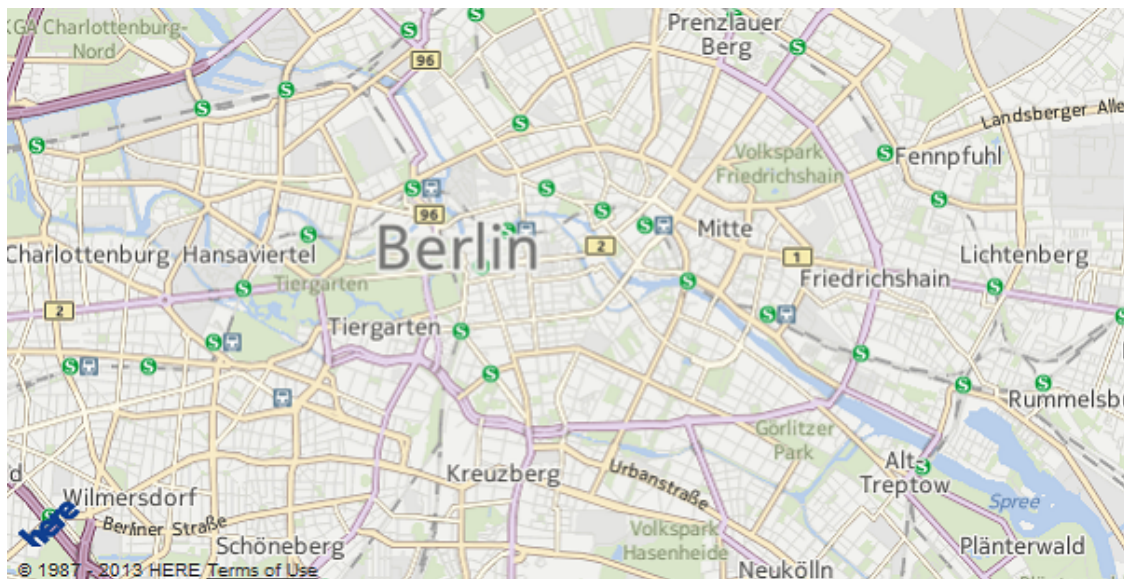
Displaying a Map

The simplest application using the HERE Maps API for JavaScript is one that displays a non-interactive map centered on a predefined location at a fixed zoom level. To implement it:

1. Define a target HTML element in which the map is to be rendered, setting the "id" attribute on it.
2. Instantiate `nokia.maps.map.Display`, passing to the constructor the id of the target element, the desired map zoom level and the coordinates of the location on which you want to center of the map

Below, you can see the outcome of implementing this simple scenario as a map image and also (further down the page) the code showing how it has been achieved.

Figure 1: A basic non-interactive map



```
var map = new nokia.maps.map.Display(  

```

```
document.getElementById("mapContainer"), {  
  // Zoom level for the map  
  zoomLevel: 10,  
  // Map center coordinates  
  center: [52.51, 13.4]  
};
```

This code places the map inside a `<div>` with the `"id"` set to `"mapContainer"`. Other DOM elements that support nesting can be used as well, for example `<span>`, `<p>`, or `<em>`, but we strongly recommend `<div>` as the host element for the map for best results.

Bear in mind that most browsers use predefined styles for many DOM elements. To ensure a consistent look and feel in different browsers, you need to apply your own CSS style to the DOM element that displays the map. The code examples shown in this guide demonstrate how to do this in a simple, but effective browser-independent way.

here

Chapter

3

User Guide

Topics:

- [Acquiring Credentials](#)
- [Beyond the Basic Map Application](#)
- [Map Types](#)
- [Components and User Interaction](#)
- [Markers](#)
- [Geo Objects](#)
- [Map Overlays](#)
- [Routing](#)
- [Traffic](#)
- [KML Support](#)
- [Heat Maps](#)
- [Marker Clustering](#)
- [Positioning](#)
- [Event System](#)
- [Observers and Mutable Objects](#)
- [Multi-language Support](#)

This part of the guide explains essential concepts and shows how to use the Maps API for JavaScript. It is a companion to the API Reference.

The text includes examples that show the API in operation. Note that where credentials are required, the examples use demo ones. These are acceptable for testing purposes, but must be replaced by application-specific credentials in real-life applications (please see also [Acquiring Credentials](#) on page 18).

The HERE Maps API for JavaScript has been designed to help developers meet the challenges of modern Web application development. The articles that follow describe various aspects of the API, including a number of state-of-the-art development patterns it supports.

Acquiring Credentials

All users of HERE APIs must obtain authentication and authorization credentials and provide them as values for the parameters `app_id` and `app_code`. The credentials are assigned per application.

To obtain the credentials for an application, please visit <http://developer.here.com/get-started> for more details.

Beyond the Basic Map Application

Quick Start on page 11 describes an elementary scenario confined to displaying a non-interactive map is shown on the screen. Real-life applications are likely to go further, using the Maps API to deliver interactive functionality, animations, etc. For example, an application may start by showing a street/physical map and allow the user to select a different map type such as satellite. The application may also zoom in on a particular location on the map in response to the user double-clicking on the map.

In the first of these examples, the application user is able to effect a change to the "base map type", while switching to a different map zoom level involves animation. As a developer, you need to consider that to display the map when the application starts and then subsequent changes to the map type or map view and any animations are asynchronous. If another operation occurs before an asynchronous changes has completed, the results may be unpredictable, with your application becoming unstable. To prevent this, your code must be listen for certain events and act on them:

- **displayready** – fired when the map is initially ready; no code related to map view manipulation, such as calls to `setZoomLevel()`, `setCenter()`, `setBaseMapType()`, etc. must be allowed to run (although you can add components or markers to the map and perform other operations that do not affect the map view).
- **basemapchangestart**, **basemapchangeend** – fired to mark the start and end of a change to the base map type; no code that must be allowed to modify the map view until `basemapchangeend` has been received, otherwise the map may display an unexpected location or it may become unstable.
- **transitionstart**, **transitionend** – fired to mark the beginning and end of a map animation (transition) and also between the events `basemapchangestart` and `basemapchangeend`, because base map changes include animation effects (if transition parameter is set to 'default'); no code that must be allowed to modify the map view until `transitionend` has been received, otherwise the map may display an unexpected location or it may become unstable.

here

The following sections show examples of program code detecting `displayready` and `transitionend`. Examples involving `basemapchangeend` are included in the article [Map Types](#) on page 20.

For information about support for other types of events in the Maps API, please see [Event System](#) on page 45.

Waiting for Display to Be Ready

The example below is very similar to the code you have seen in [Quick Start](#) on page 11, but it uses the method `addListener()` defined on the `Display` class to register a handler for the event `displayready`. The handler has an empty body, but in real life its code is the entry point to the implementation of the functionality of the application – functionality which requires the map to be shown as the first enabling step.

```
var map = new nokia.maps.map.Display(
    document.getElementById("mapContainer"), {
        // Zoom level for the map
        zoomLevel: 10,
        // Map center coordinates
        center: [52.51, 13.4]
    });

map.addListener("displayready" function () {
    // Place code implementing the functionality of app
    // here, for example to manipulate the map view.
});
```

Changing the Map View and Animation

As we have mentioned earlier, several methods defined on the Maps API class `Display` can be called to change the map view, for example `setZoomLevel()`, `setCenter()`, `setView()`. If the second argument is specified as `'default'` and the target map view is not too far from the current one, the map moves to the target view, using a transition (animation). The beginning of the transition is signaled by the event `transitionstart` and its end by `transitionend`. In the code that follows, the handler for `displayready`, changes the map zoom level from the initial 18 to 15 after a time-out of one second. It also defines a listener for the event `transitionend`. This listener is invoked when the map zoom level change has completed, merely to log that this is the case.

```
map = new nokia.map.Display(document.getElementById("mapContainer"), {
    center: [42.35086, -71.07149],
    zoomLevel: 18
});
map.addListener("displayready", function () {
    //animate zoom level change during the application run
    setTimeout(function () {

        function onTransitionFinished() {
            map.removeListener("transitionend", onTransitionFinished);
            console.log("transition finished");
        }
    }, 1000);
});
```

here

```
map.addListener("transitionend", onTransitionFinished);
    map.setZoomLevel(15, 'default');
  }, 1000);
});
```

Map Types

The Map API allows you to choose a *map type* to display. For example, your application can use an ordinary road/street map, or a satellite map (based on satellite imaging), or a terrain map. The class `Display` defines constants that represent the available base map types:

Table 4: Map types supported by the HERE Maps API for JavaScript

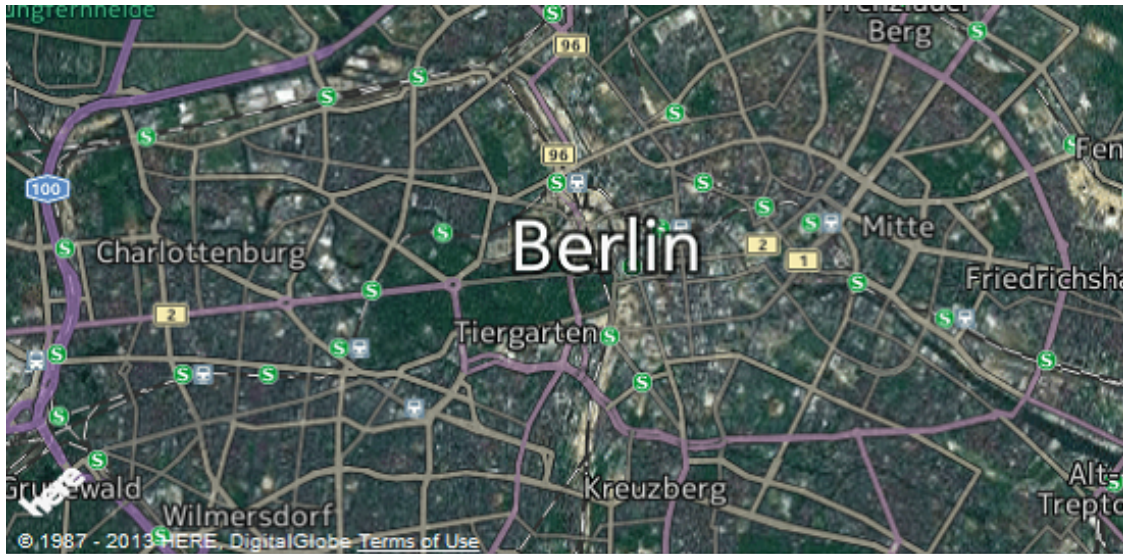
Map type	Description
NORMAL	This is the default base type showing a conventional map
SATELLITE	This is a type that offers a view of the map based on satellite imagery
TERRAIN	This is a type that shows a map based on elevation profile imagery (a topographical map, with shading indicating the shape of the terrain)
TRAFFIC	This is a type that shows a map with traffic information

To set or change the base map type, use the method `setBaseMapType( )` on an instance of the `Display` class:

```
// Activate satellite imagery on the display:
map.setBaseMapType(nokia.maps.map.Display.SATELLITE);
```

The image below shows a satellite view of the area covered by the basic map in the previous example. This is achieved simply by adding the above line of code to the script, setting the map type. You can see the complete JavaScript code below the image.

Figure 2: Map type "satellite"



```
var map = new nokia.maps.map.Display(  
    document.getElementById("mapContainer"), {  
        zoomLevel: 10,  
        center: [52.51, 13.4],  
        // Activate satellite imagery on the display:  
        baseMapType: nokia.maps.map.Display.SATELLITE;  
    }  
);
```

Changing between Map Types and Setting a New Map View

A map type sets the overall appearance of the map, but not its orientation (heading), center or tilt, which are features of the *map view*. However, when you change the map type, it may be desirable to alter the map view as well. The Maps API allows you to do that and to ensure that the transition appears smooth to the map user's eye.

To trigger animation between map types, you need to provide a second parameter to the method `setBaseMapType()`. The required value is 'default'. It ensures a smooth transition to the target map type.

The code below demonstrates how to implement a map type change to SATELLITE, with a smooth transition.

```
var map = new nokia.maps.map.Display(document.getElementById("mapContainer"), {  
    zoomLevel: 16,  
    center: [52.51, 13.4],  
    // Trigger a smooth transition to the target map type:  
    setBaseMapType: 'default',  
    baseMapType: nokia.maps.map.Display.SATELLITE;  
});
```

here

```
baseMapType: nokia.maps.map.Display.NORMAL
});

map.addListener("displayready", function () {
    // Async transition to a new map type (after 3 seconds)
    setTimeout(function(){
        map.setBaseMapType(map.SATELLITE, 'default');
    }, 3000);
});
```

The alternative is to forgo animation and allow the map to change type immediately. This requires the second argument to `setBaseMapType()` to have the value `'none'`.

If you would like to specify a new map view when changing the map type, you need to provide a further argument to `setBaseMapType()`. This third argument is an object whose properties define the details of the target map view, including the geographic coordinates of the map center, zoom level, tilt and heading. The following example shows a call to `setBaseMapType()` that sets a new map type, ensures a smooth transition effect, and defines the details of the new the map view to which to transition:

```
map.setBaseMapType(map.SATELLITE, 'default', {
    latitude: 52.51,
    longitude: 13.4,
    zoom: 16
});
```

Map type changes are asynchronous. To avoid unpredictable results, it is important that no other interaction affects the map view while the rendering engine processes the map type change: in other words, there must be no attempt to alter the map center or zoom level or trigger (and process) any other asynchronous operations, when a map type switch is in progress. To ensure that is the case, listen for two events, the first fired when the transition has started (`basemapchangestart`) and the second when it has finished (`basemapchangeend`). They are fired by the map `Display` instance. Your handlers for these events need to eliminate unwanted map interaction, ensuring a smooth and consistent transition to the target view.

The code below shows the skeleton for the handler functions. Note that we do not attempt to process the events `transitionstart` and `transitionend`, because these are fired between `basemapchangestart` and `basemapchangeend`.

```
map.addListener("basemapchangestart", function () {
    // Map type change has started -- add
    // relevant handling here.
});

// Add listener so we know when map type change is complete:
map.addListener("basemapchangeend", function () {
    // Map type change has finished -- add
    // relevant processing here.
});
```

here

Components and User Interaction

The HERE Maps API for JavaScript ships with a basic set of User Interface (UI) components with which you can enhance an application built around an interactive map. The UI components allow the application user to zoom in or out, pan the map or change its scale, change the map type, etc.

The following UI components are available with the HERE Maps API for JavaScript:

Table 5: UI components in HERE Maps API for JavaScript

Component	Description
Behavior	A collection of components that enable general user interaction with the map via the mouse or touch (the interaction types include zoom, panning, etc.)
Zoombar	A component that provides a slider control on the map, allowing the user to zoom in/out – clicking on the plus sign increases the zoom level and causes the map to show more detail, while the clicking on the minus sign has the opposite effect.
Scalebar	A component that displays a scale bar at the bottom of the map. The user can expand it to see a ruler reflecting linear distance on the map. The user can change the units of measurement between metric and imperial.
TypeSelector	A component that displays a map type switcher in the top right corner of the map.
Overview	A component that allows the map user to see what lies beyond the current map view. The overview expands from the bottom right when the user clicks or taps on the Overview icon.
ZoomRectangle	A component that allows the map user to zoom the map to a specific rectangle (area) by dragging.
Positioning	A component that provides a button with which the map user can trigger a positioning request. On success, the map is centered on the obtained location. The size of a circle drawn around the location indicates the accuracy of the positioning data. The component is part of the positioning package.
ContextMenu	A component that shows a customized context menu on a right-click (or a longpress on touch devices).
InfoBubbles	A component that allows a developer to show information bubbles containing HTML content and to manipulate them.
PublicTransport	A component that provides a map overlay to display public transport lines. The map user can toggle the overlay, but note that the display of public transport information is confined to a range of zoom levels specific to the location/city.

Component	Description
Traffic	A component that displays traffic information on the map as an overlay. The map user can toggle the overlay, but note that the display of traffic information is confined to a range of zoom levels specific to the location/city.
TrafficIncidents	A component that displays traffic incidents on the map as markers. Note that the display of traffic information is confined to a range of zoom levels specific to the location/city.

To add the UI components in an application, you need to add them to the Display instance:

```
var map = new nokia.maps.map.Display(  
    document.getElementById("mapContainer"), {  
        components: [  
            // Behavior collection  
            new nokia.maps.map.component.Behavior(),  
            new nokia.maps.map.component.ZoomBar(),  
            new nokia.maps.map.component.Overview(),  
            new nokia.maps.map.component.TypeSelector(),  
            new nokia.maps.map.component.ScaleBar() ],  
        zoomLevel: 10,  
        center: [52.51, 13.4]  
    }  
);
```

The following is a screen shot of an application similar to that in the previous example, but it includes UI components: in fact one of them has been used to change the map type to "SATELLITE" (via one of the buttons in the top right). The image also shows other components:

Figure 3: A map with UI components supported by the HERE Maps API for JavaScript



Note that the Behavior component added at Display instantiation in the code above includes a number of other components. If you find that you do not need all of them, you can remove them by adding a statement such as the following after you have created an instance of Display:

```
var map = new nokia.maps.map.Display(
    document.getElementById("mapContainer"), {
        components: [
            // Behavior collection
            new nokia.maps.map.component.Behavior(),
            new nokia.maps.map.component.ZoomBar(),
            new nokia.maps.map.component.Overview(),
            new nokia.maps.map.component.TypeSelector(),
            new nokia.maps.map.component.ScaleBar() ],
        zoomLevel: 10,
        center: [52.51, 13.4]
    }
);

map.removeComponent(map.getComponentById("zoom.MouseWheel"));
```

This piece of code above removes zooming with the mouse wheel. This makes sense when the map application is part of a long page and you want the user to be able to scroll the page by turning the wheel, without causing the map zoom to change at the same time.

As an alternative to adding a group of components and then removing some of them, consider creating an instance of Display, and then adding only those subcomponents of Behavior that you need. The following code sample demonstrates this:

```
var map = new nokia.maps.map.Display(
    document.getElementById("mapContainer"), {
        // Zoom level for the map
        'zoomLevel': 10,
        // Map center coordinates
        'center': [52.51, 13.4]
    }
);

map.addComponent( new nokia.maps.map.component.zoom.DoubleClick());
map.addComponent( new nokia.maps.map.component.panning.Drag());
map.addComponent( new nokia.maps.map.component.panning.Kinetic());
```

See also the code example in the article [Markers](#) on page 25, where only the DoubleClick zoom and the DragMarker behavior components are enabled during instantiation of the map Display class.

Markers

One of the most common use cases is to show points of interest (POIs) at and near a location. With the HERE Maps API for JavaScript you can use markers to do just that. The API offers two sorts of markers:

here

Table 6: Marker Types in HEREMaps API for JavaScript

Marker type	Description
StandardMarker	A marker that uses standard platform icons and a predefined set of properties whose values can be controlled via the API
Marker	A marker that allows you to use a custom icon image to indicate a point on the map

In a very simple scenario, you may wish to center the map at your location and put a marker there with a text label that says "Hi!" – just as in the picture below. The code that follows the image demonstrates the implementation.

Figure 4: A map containing a draggable marker, with a custom label



```
var map = new nokia.maps.map.Display(
    document.getElementById("mapContainer"), {
        components: [
            new nokia.maps.map.component.zoom.DoubleClick(),
            // Needed for marker drag
            new nokia.maps.map.component.objects.DragMarker(),
        ],
        zoomLevel: 15,
        center: [52.51, 13.4]
    });

// Create a marker and add it to the map
var marker = new nokia.maps.map.StandardMarker([52.51, 13.4], {
    text: "Hi!", // Small label
    draggable: true // Make the marker draggable
});

map.objects.add(marker);
```

The code above creates a map and puts the marker at its center. The marker is an instance of the class `StandardMarker`, which supports custom labels via the property 'text'. This marker is also

draggable, which means you can move it around the map by clicking on it, holding the mouse button down and moving the mouse: the marker follows the mouse pointer and when you release the mouse button, it drops onto the map at a new location. This works on touch devices as well.

The Marker class in the Maps API is similar to StandardMarker, but instead of supporting custom colors and a text label, it allows you to use a custom marker icon. The image below shows just such a marker. The implementation code follows the picture.

Figure 5: A map containing a non-draggable marker



```
var map = new nokia.maps.map.Display(
    document.getElementById("mapContainer"), {
        components: [new nokia.maps.map.component.Behavior()],
        zoomLevel: 13,
        center: [52.51, 13.4]}
);

map.removeComponent(map.getComponentById("zoom.MouseWheel"));

var marker = new nokia.maps.map.Marker(
    new nokia.maps.geo.Coordinate(52.51, 13.4), {
        title: "marker",
        visibility: true,
        icon: "images/pic_marker_house.png",
        // Offset the top left icon corner so that it's
        // Centered above the coordinate
        anchor: new nokia.maps.util.Point(32, 32)
    }
);

map.objects.add(marker);
```

This code follows the same pattern as the previous examples, but creates an instance of Marker with a custom icon (its URL is given by the property "icon"). Because the icon is not a standard one, it is a

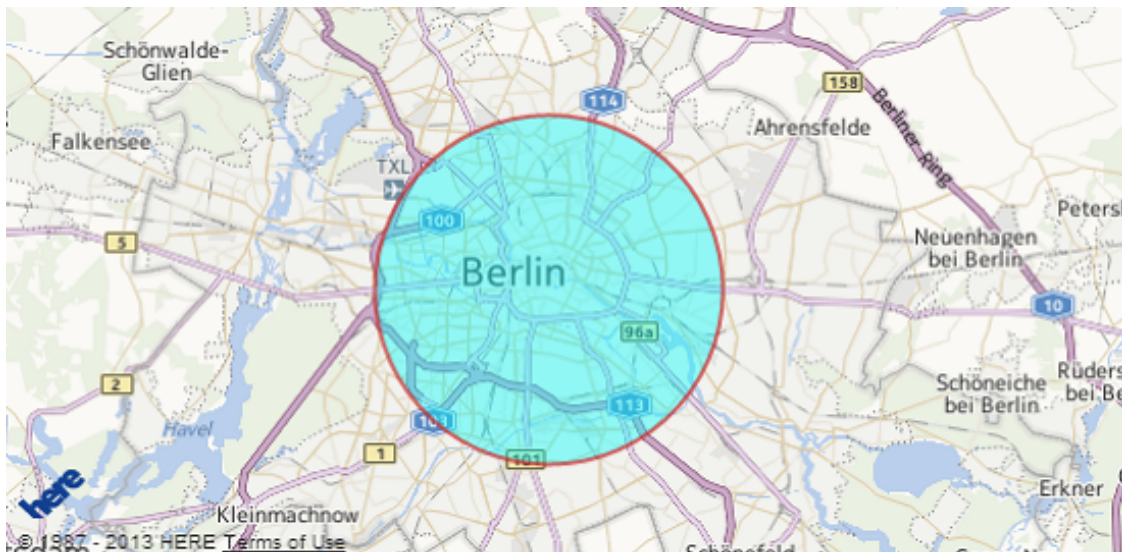
good idea to specify which point in the icon image should "touch" the map at the point defined by the marker's geographical coordinates. This is done via the property "anchor".

Geo Objects

The HERE Maps API for JavaScript provides easy means of highlighting and demarcating areas on the map with custom shapes. These include circles, rectangles, polylines and polygons. They allow you to draw on an on-line map as you can on a paper one with a pencil or pen, but with much greater flexibility, because you determine the colors, the line thickness and transparency for the graphical markings, plus you can position them with pinpoint accuracy.

The API classes that represent custom shapes need to be instantiated and added to the `Display` object. The example below demonstrates this by adding a circle to the map.

Figure 6: A map with a Circle shape



```
var map = new nokia.maps.map.Display(
    document.getElementById("mapContainer"), {
        components: [new nokia.maps.map.component.Behavior()],
        zoomLevel: 10,
        center: [52.51, 13.4]
    }
);

map.objects.add(new nokia.maps.map.Circle(
    // Place the circle center here
    [52.51, 13.4],
    // Radius of 8000 meters
    8000,
    {
        pen: {
            strokeColor: "#C22A",
```

```
        linewidth: 2
      },
      brush: {
        color: "#0FF6"
      }
    }
  ));
```

As you can see, a circle is an instance of the class named `Circle`. Its properties include the coordinates of the center, the radius, line color (defined by the property `color`), fill color and line width (given by the property `width`). All the properties of the circle can be set by passing an object (defined in JSON) to the class constructor.

Map Overlays

The Maps API allows you to display overlays on top of the map. In the simplest scenarios, the overlays are static tiles (images) that are requested by the client application from the map tile server and superimposed on the map in the client-side display. Such tiles may be images that present an alternative view of an area of the map or highlight certain features of the terrain, etc.

Other scenarios may require the use of dynamic overlays in order to improve efficiency by reducing the processing requirements on the client. For example, if an application needs to display 100,000 markers in a specific region, the designers can consider a custom map tile service to create map tiles "populated" on the fly with the required markers and deliver them to the client application on request. In another example, where a custom map tile service is likely to offer significant efficiency benefits, a transport company may wish to mark the several hundred routes it operates across the continent, using custom lines (thickness, color, etc.) Instead of requesting each route and adding it to the map, the client application can simply download the semi-transparent overlay tiles from the back-end map tile server.

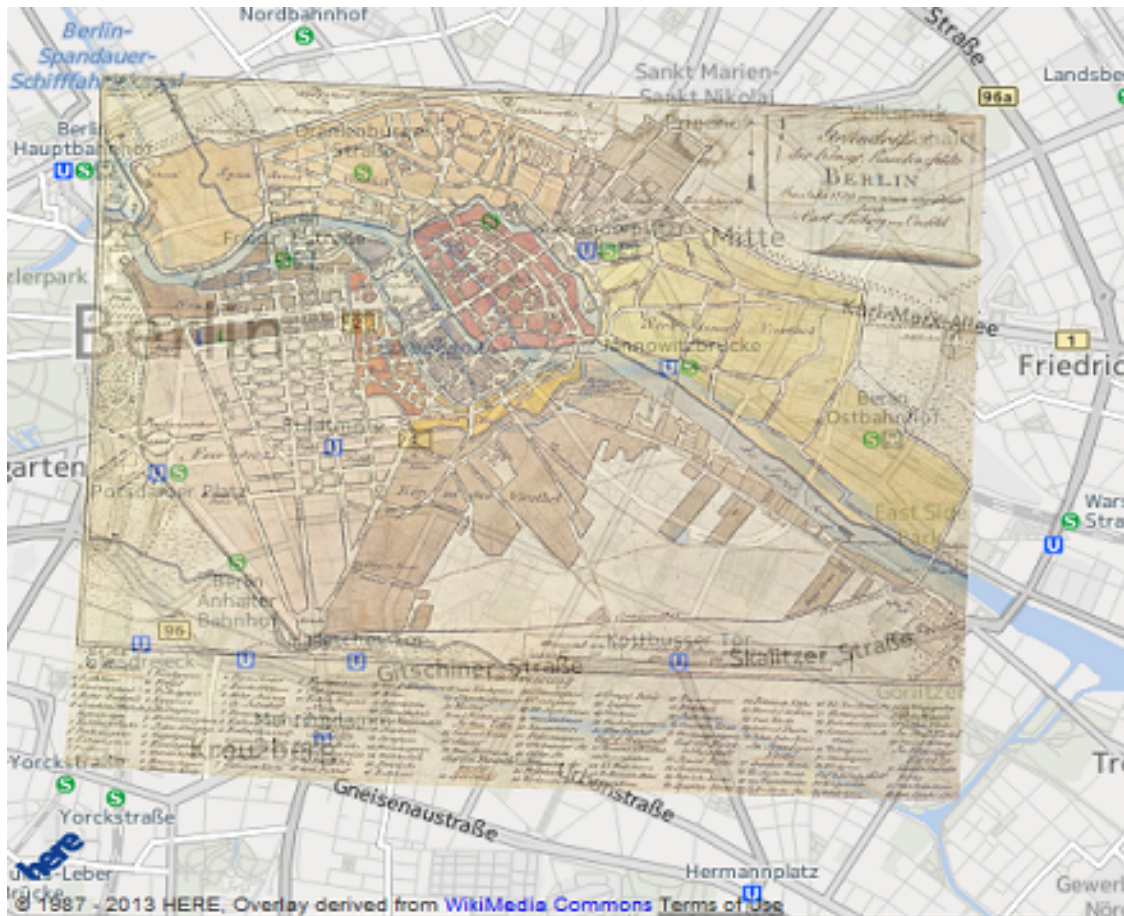
Naturally, a map tile service must be designed and implemented to specific requirements, therefore it is beyond the scope of both the Maps API and this documentation. However, we illustrate below a simple scenario in which an overlay showing a historical map of Berlin from 1789 is displayed on top of the standard modern map of the city delivered using the API.

here

A Map of Berlin from 1789

The 18th century map of Berlin in this example is based on a public domain image from [Wikimedia Commons](#). We have divided the image into tiles, which the application below retrieves from our custom map tile server and displays over a contemporary street map of the city.

Figure 7: Berlin 1789 and today



The client-side implementation, consists of the following steps:

1. Define a DOM element in which the map is to be displayed.
2. Instantiate an object representing the viewable map (`nokia.maps.map.Display`).
3. Instantiate a tile provider object (`nokia.maps.map.provider.ImgTileProvider`).
4. Add the tile provider to the array of map overlays (`nokia.maps.map.Display.overlays`).

The section below expands on each of these points.

here

Base Map

The DOM element, where the map is to appear, is defined in the body of the HTML page as follows:

```
<div id="mapContainer"></div>
```

To access the API library, provide a `<script>` element that references it:

```
<script src="http://js.api.here.com/ee/2.5.4/js1.js" type="text/
javascript" charset="utf-8"></script>
```

Set the application id and code – please see [Acquiring Credentials](#) on page 18 for details.

To display the basic street map of Berlin, get a reference to the DOM element that is to display it and instantiate the map Display:

```
// Get the DOM node to which to append the map
var mapContainer = document.getElementById("mapContainer");

// Create a map inside the map container DOM node
var map = new nokia.maps.map.Display(mapContainer, {
    components: [
        new nokia.maps.map.component.ZoomBar(),
        new nokia.maps.map.component.Behavior(),
        new nokia.maps.map.component.TypeSelector(),
    ],
    zoomLevel: 14,
    center: [52.515, 13.405]
});
```

Tile Provider

To create a tile provider, it is necessary to instantiate the base class `nokia.maps.map.provider.ImgTileProvider`. In fact, this is the most complex part of the process. The constructor for `ImgTileProvider` requires an object described in the API Reference as `nokia.maps.map.provider.TileProvider.Options`. Only one element in `Options` is obligatory, a function that retrieves the overlay tiles (`getUrl`), but the code below specifies also non-mandatory properties, including the minimum and maximum zoom levels for the overlay, opacity (which makes the overlay semi-transparent) and a function that displays copyright and attribution information:

```
var tileProviderOptions = {
    getUrl: getTileURL, // obligatory
    max: 15, // max zoom level for overlay
    min: 12, // min zoom level for overlay
    opacity: 0.5, // 0 = transparent overlay, 1 = opaque
    alpha: true, // renderer to read alpha channel
    getCopyrights : showCopyright // display copyright
};
```

It is necessary to provide a value for the property `getUr l`, otherwise the Maps API library generates an error message and terminates execution of the code. In our case, the value is the function `getTileURL ( )`, which retrieves the tiles from the map tile server. The following principles apply:

1. At the minimum zoom level, the world is contained in a single 256x256 pixel tile.
2. At the next higher zoom level, the world is two tiles wide and two tiles high.
3. Every higher zoom level splits the tiles in two, doubling the number of tiles across the width and height. This produces the tile grids of 4x4, 8x8, 16x16, etc.
4. Typically, each map is based on the Normalized Mercator projection.

Central Berlin is displayed in tile 1343,2200 at Zoom 12. The range of available tiles doubles at each zoom level thereafter.

Here is the implementation of `getTileURL ( )`:

```
function getTileURL (zoom, row, column) {
    // If Berlin is not displayed, return a blank
    // tile.
    if (((zoom == 12) && ( row != 1343 || column != 2200)) ||
        ((zoom == 13) && (row < 2686 || column < 4400 ||
        row > 2687 || column > 4401 )) ||
        ((zoom == 14) && ( row < 5372 ||
        column < 8800 || row > 5375 || column > 8803)) ||
        ((zoom == 15) && ( row < 10744 ||
        column < 17601 || row > 10750 || column > 17607))) {
        return "." + "oldberlin/blank.png";
    }

    // The Old Berlin Map Tiler follows the TMS URL
    // specification. According to the specification,
    // map tiles should be accessible in the following
    // format:
    //
    // http://server_address/zoom_level/x/y.png
    //
    // In our case, the URL is something like
    // "oldberlin/13/1343/2200.png"
    return "." + "oldberlin/" + zoom + "/" + row + "/" +
        column + ".png";
};
```

Unless you own the map tile source, you need to comply with the licensing agreement of the map tile provider. Often, this means giving an attribution text or acknowledgement of the copyright of the original owner, even if the image tiles are offered free of charge. The function `showCopyright ( )` function helps display this acknowledgement in the bottom left corner of the map.

```
function showCopyright (area, zoom) {
    // If the zoom is too low or too high, do not display any
    // copyright information:
    if (zoom < 12 || zoom > 15) {
        return [];
    }

    // Otherwise, add the attribution to the map:
    return [{
        label: "Overlay derived from <a href="
```



```
'http://commons.wikimedia.org/wiki/File:Map_de_berlin_1789_%28georeferenced%29.jpg'>
  Wikimedia Commons</a>",
  alt: "Overlay Based on a Wikimedia Commons Image in the Public Domain"
}
}];
}
```

Displaying the Overlay

With the code above in place, we can now create an instance of the tile provider:

```
var oldBerlinOverlay =
  new nokia.maps.map.provider.ImgTileProvider (
    tileProviderOptions);
```

Finally, we can add the overlay to the map (and so display it):

```
map.overlays.add(oldBerlinOverlay);
```

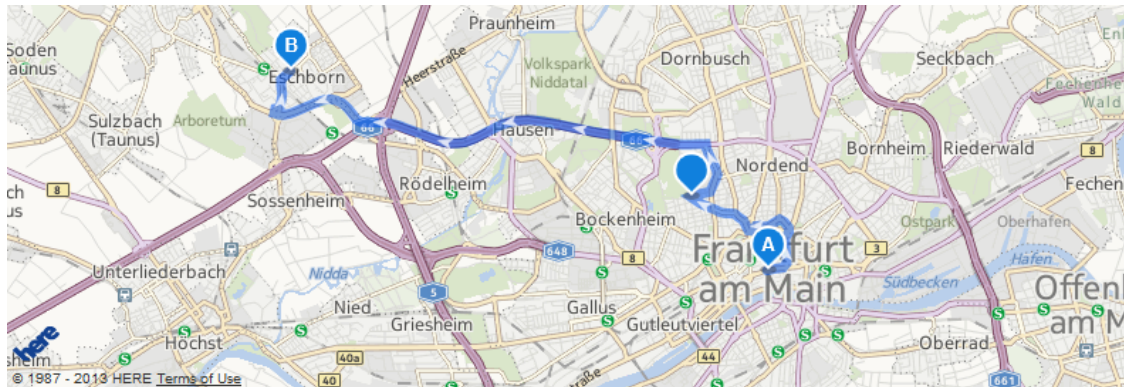
Routing

Route planning and navigation are the most commonly used applications of location-based services. With the Maps API, you can calculate optimal routes that match your own calculation criteria, are based on up-to-date map data, and take into account real-time traffic information.

The API offers global coverage of streets and highways, allowing you to create routes reflecting customizable modes such as fastest, shortest, avoiding toll roads or ferries, etc. There is also support for utilizing historical speed patterns as an optimization for routes depending on the time of day.

The image below shows a route between Frankfurt am Main in Germany (marker A) and the town of Eschborn (marker B). It is followed by code that shows the main points of the implementation.

Figure 8: A map showing a driving route from Frankfurt am Main to Eschborn



To create a route display like this, let us begin by defining a number of variables:

- A reference to the DOM element in which the map (and the route) is displayed
- Two rectangular areas which the route is to avoid
- The start and destination for the route, plus one point through which the route is to pass
- An object containing the parameters that define the route (including the start and destination, areas to avoid, etc.)
- A reference to the map Display object to which the route can be added

These variables and their definitions are shown in the code block below:

```
// Get the DOM node to which to add the map -- it presupposes
// that a DOM element with the id "mapContainer" exists,
// for example <div id="mapContainer"></div>:
var mapContainer = document.getElementById("mapContainer"),
// First area for the route to avoid
bbox1 = new nokia.maps.geo.BoundingBox(
    new nokia.maps.geo.Coordinate(50.1117, 8.6714),
    new nokia.maps.geo.Coordinate(50.1091, 8.6813)
),
// Second area for the route to avoid
bbox2 = new nokia.maps.geo.BoundingBox(
    new nokia.maps.geo.Coordinate(50.1208, 8.664),
    new nokia.maps.geo.Coordinate(50.1182, 8.6726)
),
// Departure point for the route
pointA = new nokia.maps.geo.Coordinate(50.112, 8.6834),
// An intermediate waypoint for the route
pointB = new nokia.maps.geo.Coordinate(50.1431, 8.5680),
// Destination point for the route
pointC = new nokia.maps.geo.Coordinate(50.1226, 8.6644),
routingRequestParams = {
    waypoints:[
        {position: pointA},
        {position: pointC, transitRadius:100},
        {position: pointB}
    ],
    avoidArea:[bbox1,bbox2],
    departure: (new Date()).toISOString(),
```

```

alternatives: 2,
metaInfo: {requestId: "sampleRequest"},
representationOptions: {
  language: "en-US",
  representationMode: "overview",
  /* possible routeAttribute switches:
   * "waypoints(wp)", "summary(sm)",
   * "summaryByCountry(sc)", "shape(sh)",
   * "boundingBox(bb)", "legs(lg)",
   * "notes(no)"
   */
  routeAttributes: ["wp", "sc", "sm", "sh", "bb", "lg", "no"],
  /* possible legAttribute switches:
   * "waypoint(wp)", "maneuvers(mn)", "links(li)",
   * "length(le)", "travelTime(tt)"
   */
  legAttributes: ["wp", "mn", "li", "le", "tt"],
  /* possible maneuverAttribute switches:
   * "position(po)", "shape(sh)", "travelTime(tt)",
   * "length(le)", "time(ti)", "link(li)",
   * "publicTransportLine(pt)", "platform(pl)",
   * "equipment(eq)", "lane(la)", "roadName(rn)",
   * "nextRoadName(nr)", "roadNumber(ru)",
   * "nextRoadNumber(nu)", "roadTemplate(rt)",
   * "signPost(sp)", "notes(no)", "action(ac)",
   * "direction(di)", "nextManeuver(nm)",
   * "freewayExit(fe)", "freewayJunction(fj)"
   */
  maneuverAttributes: ["po", "sh", "tt", "le",
    "ti", "li", "pt", "pl", "eq", "la", "rn",
    "nr"
  ],
  /* possible linkAttribute switches:
   * "shape(sh)", "length(le)", "speedLimit(sl)",
   * "dynamicSpeedInfo(ds)", "incidents(ic)",
   * "truckRestrictions(tr)", "externalResources(er)",
   * "flags(fl)", "address(ad)", "roadNumber(rn)",
   * "roadName(ro)", "freewayExit(fe)",
   * "freewayJunction(fj)", "timezone(tz)",
   * "corridorLevel(cl)", "nextLink(nl)", "stubs(st)",
   * "publicTransportLine(pt)", "TMCCodes(tm)",
   * "jamFactor(jf)", "jamFactorTrend(jt)",
   * "confidence(co)", "remainTime(rt)",
   * "remainDistance(rd)", "maneuver(ma)",
   * "functionalClass(fc)", "nextStopName(ns)",
   * "additionalData(dd)"
   */
  linkAttributes: ["sh", "le", "sl", "ds", "ic", "tr", "er"],
  corridorDepth: 2,
  instructionFormat: "html"
},
modes: [{
  /* Possible routingType switches:
   * "fastest", "shortest", "economic", "scenic",
   * "fastestNow", "directDrive"
   */
  type: "shortest",
  /* Possible TransportMode switches:
   * "car", "pedestrian", "publicTransport", "truck",
   * "bicycle"
   */
  transportModes: ["truck"],
  /* Possible trafficMode switches:
   * "enabled", "disabled", "default"
   */
  trafficMode: "default"
}]
},
// Create and display a map so that a route can be added to it:
display = new nokia.maps.map.Display(mapContainer, {

```

```
center: [53, 13],
zoomLevel: 3,
components: [new nokia.maps.map.component.Behavior()]
});
```

Please note that the comments in the definition of the `routingRequestParams` object above list a number of alternative property values.

Next, we need to create an instance of the routing manager, and save a reference to it in a variable:

```
var routingManager = new nokia.maps.advrouting.Manager();
```

A route can be added to the map only after it has been calculated. For this reason, the routing manager must be given an observer to be called when the route is ready. The observer watches the Manager's property "state":

```
function routingCallback (observedRouter, key, value) {
    if (value == "finished") {
        routes = observedRouter.getRoutes();

        // Remove any existing route from the map:
        if (mapRoute)
            display.objects.remove(mapRoute);

        // Create the default map representation of a route
        // -- get the first route from the
        // result set:
        var mapRoute = new
            nokia.maps.routing.component.RouteResultSet(
                routes[0]).container;

        // Add the first route to the map (specifically to
        // the map objects container):
        display.objects.add(mapRoute);

        // Zoom to the bounding box of the route
        display.zoomTo(mapRoute.getBoundingBox(),
            false, "default");
    } else if (value == "failed") {
        alert("The routing request failed.");
    }
};
```

Finally, here is the function that kicks off the process by requesting a route calculation to be performed:

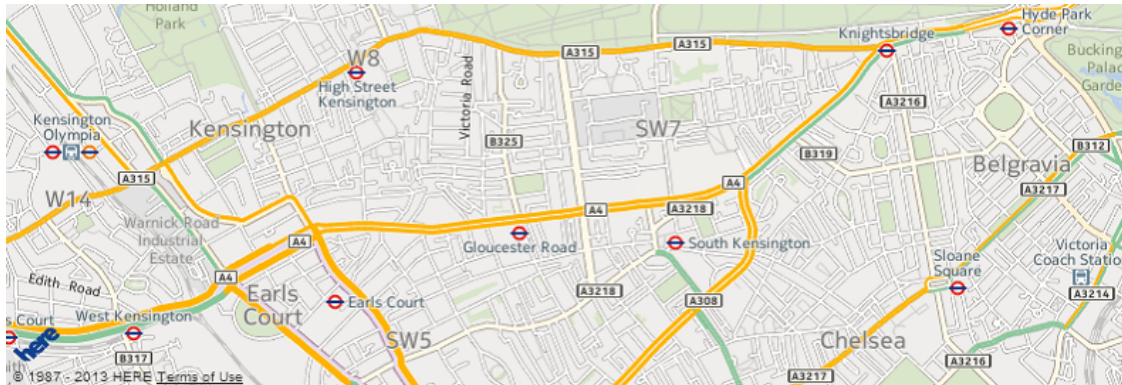
```
// Request a route calculation:
routingManager.calculateRoute(routingRequestParams);
```

Traffic

The Maps API allows you to display traffic data on the map for major urban areas around the globe. In fact, the Traffic UI component of the Maps library, provides a button that allows the user to toggle the display of traffic-related data.

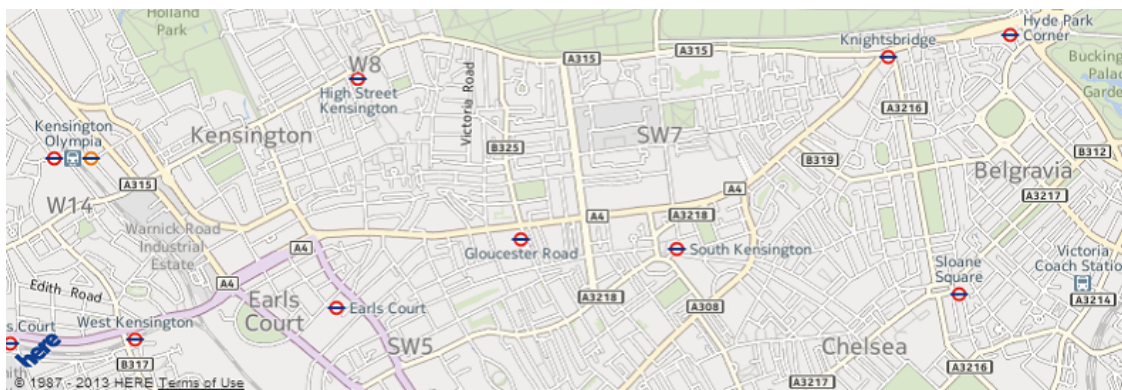
The example described below uses the traffic component of the Maps library to show a map of central and west London, indicating the traffic conditions: roads where traffic flows freely are marked in green, roads where congestion is moderate are in orange, while those that are congested are marked red.

Figure 9: A map reflecting road congestion in central London



And here is the same area of London after the user has clicked on the Traffic toggle button – the map is an ordinary street map of this area of the city.

Figure 10: A map with traffic display turned off



The implementation is very simple. The `Display` constructor's initialization object includes the properties that set the map center and zoom level and, in addition, a property named `components` to ensure that the Traffic UI component is initialized. The last line of code ensures that the initial map type is set to "traffic".

```
// Get the DOM node to which we will append the map
var mapContainer = document.getElementById("mapContainer");

// Create a map inside the map container DOM node
var map = new nokia.maps.map.Display(mapContainer, {
  // initial center and zoom level of the map
  center: [51.49, -0.12],
  zoomLevel: 13,
  components: [
    // Create a button to toggle traffic information on the map
    new nokia.maps.map.component.Traffic()
  ]
});

map.set("baseMapType", nokia.maps.map.Display.TRAFFIC);
```

KML Support

Because creating content on top of a map is time consuming, most developers look to standards to help them transfer content between platforms seamlessly, and typically use KML, which captures data in XML. For further information, please visit the [KML page of the Open Geospatial Consortium](#).

The KML specification supports a variety of objects, such as place marks, images and polygons. Many of them have their counterparts in the HERE Maps API for JavaScript landscape. To translate KML objects to HERE Maps API objects, use the component `KMLResultSet`, which is part of the `kml` package. It uses a dedicated `Manager` class to handle asynchronous loading of KML data either from a text source or from a KML file.

The HERE Maps API supports the import of KML files and data sets compliant with versions 2.1 and 2.2 of the KML standard. There are only a small number of features that the API does not as yet

support, for example, 3D buildings. The data are, however, interpreted in the best way possible and unsupported features are ignored in the KML data structure.

Figure 11: KML data rendered on a satellite map



For information on how to use the KML import in Web applications and how to interact with the KML objects, please look at the [KML file format](#) example in the [API Explorer](#).

Please note that to satisfy the same-domain policy guidelines enforced by modern Web browsers, our example file and the application are hosted in the same domain, otherwise your browser would most likely raise a security exception.

Heat Maps

In certain circumstances, you may wish to show geographic information on the map in a way that reflects the frequency of occurrence of locations with specific characteristics or the relative values associated with locations. Markers, circles, polygons or polygons are useful when you simply want to show where places or areas are, but to indicate of relative values such as occurrence or intensity, it is better to use heat maps. To cater for such scenarios, the HERE Maps API for JavaScript supports two kinds of heat maps:

Table 7: Heat map types in HERE Maps API for JavaScript

Heat map type	Description
Value based	colors represent values associated with data points on the map



Heat map type	Description

Density based	colors represent the density of data points on the map

The Maps API allows you to display heat maps as map overlays.

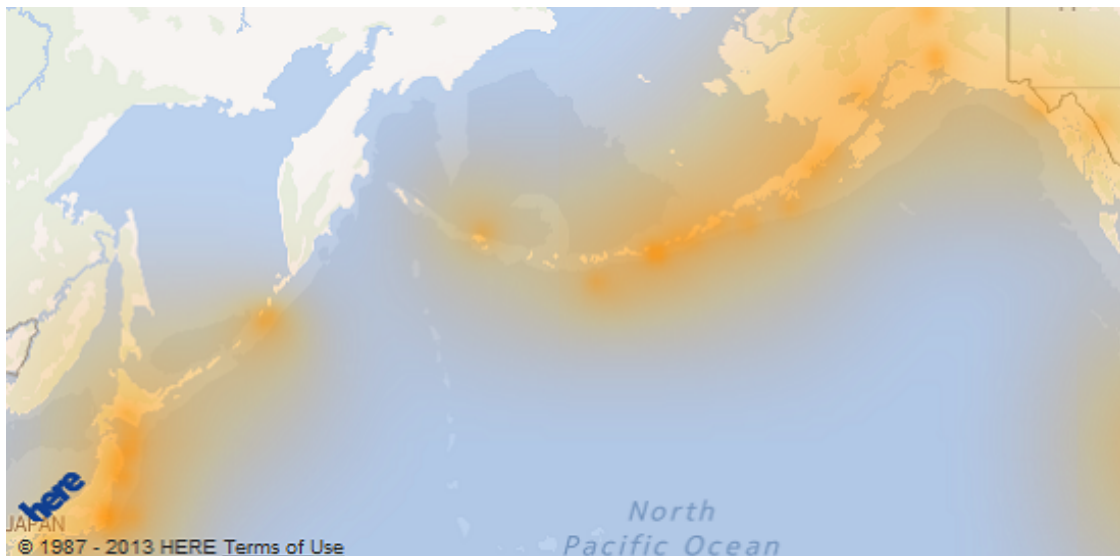
Data for the heat map are defined as a simple array of `DataPoint` objects:

```
var heatmapData = [  
  {"value":6.1,"longitude":173.0219, "latitude":53.1380},  
  {"value":5.8,"longitude":-171.8583, "latitude":52.0415},  
  {"value":5.4,"longitude":-169.9851, "latitude":53.3657},  
  {"value":4.6,"longitude":-169.5266, "latitude":51.2915},  
  {"value":4.4,"longitude":-176.4482, "latitude":51.5722},  
  {"value":4.3,"longitude":-171.5867, "latitude":51.8108},  
  {"value":4.1,"longitude":-151.8272, "latitude":59.8977},  
  {"value":3.6,"longitude":-171.7213, "latitude":51.6348},  
  {"value":3.8,"longitude":-156.0880, "latitude":56.1681}  
];
```

Note that the property "value" is optional in a density-based heat map.

The image below shows a map with a heat map overlay reflecting the density of earthquakes in Alaska based on the above data set. The image is followed by the implementation code.

Figure 12: Density based heat map



Note that heat map overlays cannot be displayed in Internet Explorer 7 or 8, because these browsers do not support the HTML5 Canvas.

The implementation of the heat map shown in the code below demonstrates how to add a heat map overlay to a map. The first step is to obtain an instance of the map `Display`. Next, the code defines an object that conforms to the interface `Colors`, setting the heat map gradients. After that, the

code constructs a heat map overlay object, adds data to the overlay, and finally, adds the overlay to the map.

```
var map = new nokia.maps.map.Display(document.getElementById("mapContainer"), {
    zoomLevel: 3,
    center: [57.0415, -170.8583]
});

// Definition of color gradient to be used in the Heatmap
var colorizeAPI = {
    stops: {
        "0": "#E8680C",
        "0.25": "#F5A400",
        "0.5": "#FF9000",
        "0.75": "#FF4600",
        "1": "#F51F00"
    },
    interpolate: true
};

var hmProvider = new nokia.maps.heatmap.Overlay({
    colors: colorizeAPI,
    opacity: 0.8,
    type: "density"
});

// Assuming that data have been loaded previously:
hmProvider.addData(heatmapData);
map.overlays.add(hmProvider);
```

Marker Clustering

When the map needs to show a very large number of marked locations, for example, to indicate stores that belong to a popular supermarket chain with thousands of branches in a particular region, it is easy to imagine that at certain zoom levels, the icons indicating the locations can appear so close together that they overlap and even hide a whole area of the map. If the locations were indicated, using individual markers, the map could become difficult to read, unless the map user zooms in to (or near) street level. To assist in managing such scenarios, the Maps Enterprise API for JavaScript includes the namespace `nokia.maps.clustering`. It offers a means of displaying a dense cluster of markers as a single marker, which is replaced by individual markers as the map user zooms in.

The key classes in the namespace are `Cluster`, `Noise`, `ClusterProvider`, and `MarkerTheme`. As the name suggests, `Cluster` represents a cluster of markers and contains references to its members. `Noise` encapsulates individual markers that are not clustered. `ClusterProvider` is a cluster manager that uses an instance of `MarkerTheme` to display clusters and single markers (instances of `Noise`). You can view the detailed documentation of each of these classes in the API Reference. It is worth noting, however, that `ClusterProvider` can work both with the default display theme (an instance of `MarkerTheme`) and with a custom theme that can use marker images of your choice.

here

The default display theme shows a cluster on the map as a circular marker with a number in its center to indicate the size of the cluster. The color, size and shadow of the cluster marker depend on the size of the cluster. It displays Noise points as small blue dots, with white stroke color. In a custom theme, the markers, their appearance and behavior depend on the implementation.

The remainder of this article describes a simple example of a clustering application that uses the default display theme. The application displays the location of 6978 airports on the map, their locations held in an array formatted as follows:

```
/**
 * Original source: http://openflights.org/data.html
 */
var data = [
  { latitude: -6.081689, longitude: 145.391881 },
  { latitude: -5.207083, longitude: 145.7887 },
  { latitude: -5.826789, longitude: 144.295861 },
  ...
];
```

To implement clustering based on this data, the example application:

1. Creates an instance of `nokia.maps.map.Display`
2. Creates an instance of `ClusterProvider` – its constructor requires a reference to the `map.Display` object as well as an object holding initialization properties for the default instance of `MarkerTheme`
3. Loads the script file containing the airport location data
4. Invokes the method `cluster()` on the `ClusterProvider` object

Below, you can see the complete contents of a `<script>` tag implementing the application. The code comments explain the details at each step:

```
// Get the DOM node in which to display the map. (This code
// assumes the presence of an HTML <div> element with
// the id "mapContainer".
var mapContainer = document.getElementById("mapContainer");

// Display a map inside the DOM node to which mapContainer
// refers:
var map = new nokia.maps.map.Display(mapContainer, {
  // initial center and zoom level of the map
  center: [52.51, 13.4],
  zoomLevel: 4,
  components: [
    // ZoomBar provides a UI to zoom the map
    // in/out
    new nokia.maps.map.component.ZoomBar(),
    // Add the component that allows user to
    // pan and zoom
    new nokia.maps.map.component.Behavior()
  ]
});

/*
 * Create an instance of ClusterProvider. It uses the default
 * display theme (an instance of
 * nokia.maps.clustering.MarkerTheme).
 * The code sets the epsilon distance (the radius within
```

here

```

    * which data points are considered for clustering, measured
    * in pixels), the smallest number of points within the
    * epsilon radius required to form a cluster, and an empty
    * array representing the data points (the actual data points
    * are loaded dynamically).
    */
    var ClusterProvider = nokia.maps.clustering.ClusterProvider,
        clusterProvider = new ClusterProvider(map, {
            eps: 16,
            minPts: 1,
            dataPoints: []
        });

    /* Load the data point definitions from the file airport.js
    * and provide a callback that adds the airport location
    * data to the instance of ClusterProvider and then creates
    * the markers and clusters.
    */
    loadScript(
        "../resource/clustering/airports.js",
        function () {
            clusterProvider.addAll(data);
            clusterProvider.cluster();
        }
    );

```

Positioning

The user's location is one of the key pieces of information in building a Web map application. The HERE Maps API for JavaScript supports the use of W3C browser positioning, making it easy for an application to use positioning information.

Precision depends on the positioning technology and ranges from rough IP Positioning (only supported in a few browsers such as Mozilla Firefox), cell positioning (usually available on tablets and notebooks) up to very precise WiFi detection (all WiFi-enabled modern browsers, among them mobile browsers). In addition, The API supports GPS or A-GPS on tablets and smart phones, which delivers very accurate positioning.

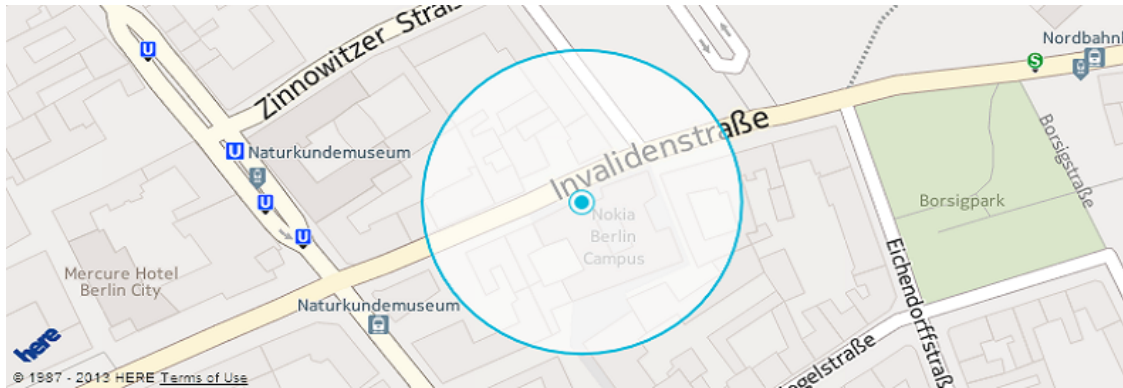
Visual presentation of positioning data on the map includes proximity rendering.

Example

The example below demonstrates the use of W3C geolocation API to obtain positioning information (location) and to display it as a marker on the map. A positioning request is submitted if the application user's mouse hover over the map. On success, a message confirms this and, in addition, the map displays a marker in a circle that reflects the accuracy of the positioning information. On

failure, the application displays an error message. The image below shows the success scenario, with a marker and the accuracy circle marking the detected location.

Figure 13: Using the W3C geolocation API (only works in browsers that support this functionality)



As in all the previous examples, the code creates an instance of `map Display`, but it also obtains an object representing the positioning Manager. It also defines a callback function named `getPos()`, which obtains the current geographic position from the browser and then creates a `StandardMarker` plus the accuracy circle and sets the map zoom level to ensure the entire circle is visible. The callback is associated with the "mouseover" event for the HTML element that contains the map.

```
var map = new nokia.maps.map.Display(
    document.getElementById("mapContainer"), {
        components: [new nokia.maps.map.component.Behavior()],
        zoomLevel: 13,
        center: [52.51, 13.4]
    });

if (nokia.maps.positioning.Manager) {
    var positioning = new nokia.maps.positioning.Manager();

    // Get the current position. If available, the first callback is run,
    // otherwise the second.
    positioning.getCurrentPosition(
        function (position) {
            var coords = position.coords;
            var marker =
                new nokia.maps.map.StandardMarker(coords);
            var accuracyCircle =
                new nokia.maps.map.Circle(coords, coords.accuracy);
            map.objects.addAll([accuracyCircle, marker]);
            map.zoomTo(accuracyCircle.getBoundingBox(), false, "default");
        },
        // Handle errors (display message):
        function (error) {
            var errorMsg = "Location could not be determined: ";

            // Determine what caused the error and show error message:
            if (error.code == 1)
                errorMsg += "PERMISSION_DENIED";
            else if (error.code == 2)
                errorMsg += "POSITION_UNAVAILABLE";
            else if (error.code == 3)
```

```

        errorMsg += "TIMEOUT";
    else errorMsg += "UNKNOWN_ERROR";
    alert(errorMsg);
};
    )
}

```

Event System

The HERE Maps API for JavaScript offers its own standardized event framework to handle user interactions such as mouse clicks and dragging. Because different browsers handle events differently, the framework harmonizes those differences in a way that is compliant with W3C standards and gives the developer an effective tool to process events on native DOM nodes. The framework also offers support for de facto standard events such as touch and gestures in the environments that support them, and it can be extended to apply to native DOM nodes.

All the classes related to the event framework are defined within the `dom` namespace. This includes the `Page` class, which defines an abstraction level for native browser events and is, in fact, the entry point to the functionality offered by the package. The namespace `dom` also includes the helper classes `Event`, `EventTarget` and `DataTransfer`.

The signatures for the event functions, for example those that allow you to add listeners, do not comply strictly to the W3C specifications, but are compatible with other JavaScript frameworks.

The supported events are listed and described in the table below:

Table 8: Events in HERE Maps API for JavaScript

Event	Description
<code>focus</code>	Fired if an element receives focus.
<code>blur</code>	Fired if an element loses focus.
<code>mousedown</code>	Fired if a mouse button has been pressed.
<code>mouseup</code>	Fired if a mouse button has been released.
<code>mousemove</code>	Fired if the mouse is moved.
<code>mousewheel</code>	Fired if the mousewheel is moved.
<code>click</code>	Fired after a <code>mousedown/mouseup</code> or a <code>touchstart/touchend</code> has occurred without a drag operation having been started. For <code>touchstart</code> and <code>touchend</code> , the <code>click</code> event is simulated if no tap listener is registered or if the tap listener has not prevented the default by using the <code>preventClick()</code> method.
<code>dblclick</code>	Fired after two <code>click</code> events have been fired within a certain time period.
<code>mouseover</code>	Fired if the mouse is moved above a new target.

Event	Description
mouseout	Fired if the mouse left a target.
mouseenter	Fired if the mouse is moved above a new target.
mouseleave	Fired if the mouse leaves the outer bounding box of a target for which a <code>mouseenter</code> has been fired.
dragstart	Fired at an event target that has the property <code>"draggable"</code> set to <code>true</code> and after a <code>mousedown/touch</code> has occurred and the mouse/finger has been moved by at least three pixels. If the event is not canceled, then the drag operation is permitted. The property <code>"dataTransfer"</code> can be used to keep track of information while dragging.
drag	Fired at the target node of the <code>dragstart</code> event while a drag operation is ongoing. The <code>"relatedTarget"</code> property contains a reference to the target that is currently under the mouse point or the user's finger tip (only for simulated drag events, not for the native ones).
dragenter	Fired at a node if the mouse/finger is moved into the visible area of the element, during a drag operation. If the event is canceled, a drop at the target is allowed, otherwise a drop is denied. If the drop is allowed, the target becomes the current drop target.
dragover	Fired at the current drop target while the mouse/finger is above the drop target, but only after the <code>dragenter</code> event has been canceled. If this event is not canceled, the drop effect is set to <code>"none"</code> , which disallows a drop into the drop target.
dragleave	Fired at a node, when the mouse/finger has been moved away from it.
drop	Fired if the mouse button or finger is released above a valid drop target. It means that the <code>dragenter</code> event and the <code>dragover</code> event have been canceled and the allowed effect matches the drop effect. If the event is not canceled and the drop target is a text field (for example a text area or an input element), then the content of the <code>"text/plain"</code> format is inserted into this area (either at the cursor position or at the end). Otherwise, the drop effect is set to <code>"none"</code> .
dragend	Fired at the end of a drag operation at the original <code>dragstart</code> target, both if the drop is aborted or failed, or if the drop was successful. The event signals the end of a drag operation. The state of the drop operation can be read from the drop effect, which can be <code>"none"</code> , <code>"move"</code> , <code>"copy"</code> or <code>"link"</code> .
touchstart	Fired as soon as a finger is pushed on a touch screen.
touchmove	Fired as soon as a finger is moved above a touch screen.
touchend	Fired as soon as a finger is released from a touch screen.
gesturestart	Fired as soon as two or more fingers touch the touch screen (high-level event).
gesturechange	Fired whenever either of two fingers is moved, changing the properties of the gesture (high-level event).
gestureend	Fired as soon as only one finger is left touching the touch screen and as soon as all fingers have been removed from the touch screen. (high-level event).
tap	Fired after a <code>touchstart</code> and <code>touchend</code> have occurred at the same target and only if neither the <code>touchstart</code> nor the <code>touchend</code> event has been canceled.
dbltap	Fired after two tap events have been fired in a certain amount of time.

Event	Description
longpress	Fired after a mouse button or finger has been pressed for a certain amount of time without starting a drag or gesture.

To add support for Page events to a DOM node, the `EventTarget` interface must be attached to the DOM node. This can be done by simply casting the node to `EventTarget`. The code implements a simple scenario in which a static map is displayed in one DOM node and a green rectangle underneath it in another. When the user clicks on the green area, the code changes the color to gray and shows an alert to indicate that it has received (and processed) the "click" event.

```
<div id="mapContainer" style="width:785px; height:460px;"></div>
<div id="eventElt" style="background-color: green; width:785px; height:20px;"></div>
<script type="text/javascript">
    var map = new nokia.maps.map.Display(
        document.getElementById("mapContainer"), {
            zoomLevel: 10,
            center: [52.51, 13.4]
        });

    // Create a few shortcuts.
    var Page = nokia.maps.dom.Page,
        EventTarget = nokia.maps.dom.EventTarget,
        eventElt = document.getElementById("eventElt"); // green rect

    // Query Page support for the node.
    Page(eventElt);

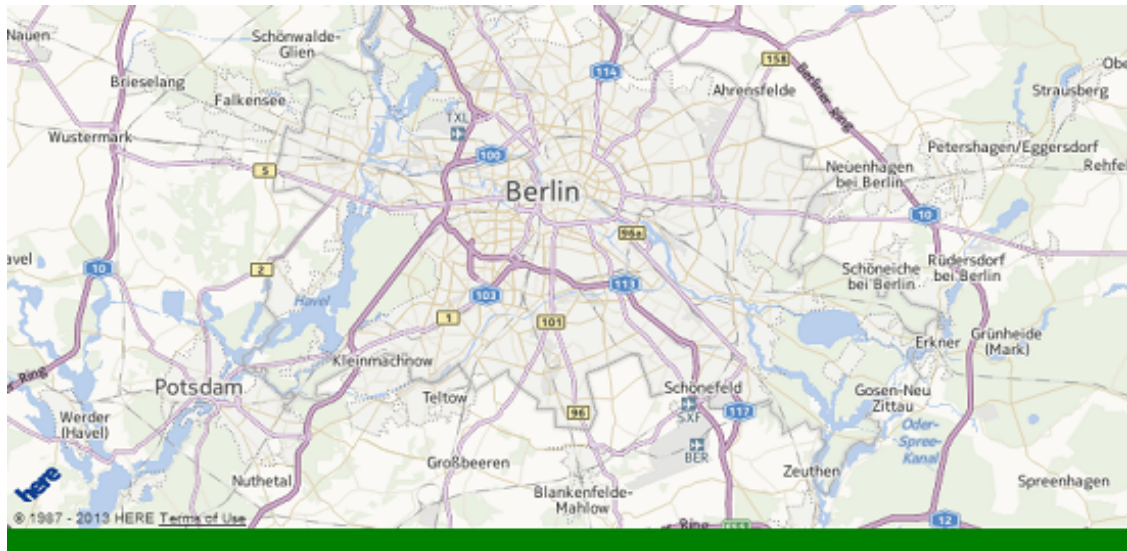
    // Attach EventTarget interface to the document node to allow
    // normalized events at the node.
    EventTarget(eventElt);

    // Add a listener for the click event to the node. It changes
    // the background color and shows an alert.
    eventElt.addListener("click", function (evt) {
        eventElt.style.backgroundColor = "#D3D3D3";
        alert( "Event triggered: " + evt.type );

        // Unregister the listener, so that the alert is shown only
        // once.
        eventElt.removeListener("click", arguments.callee, false);
    }, false);
</script>
```

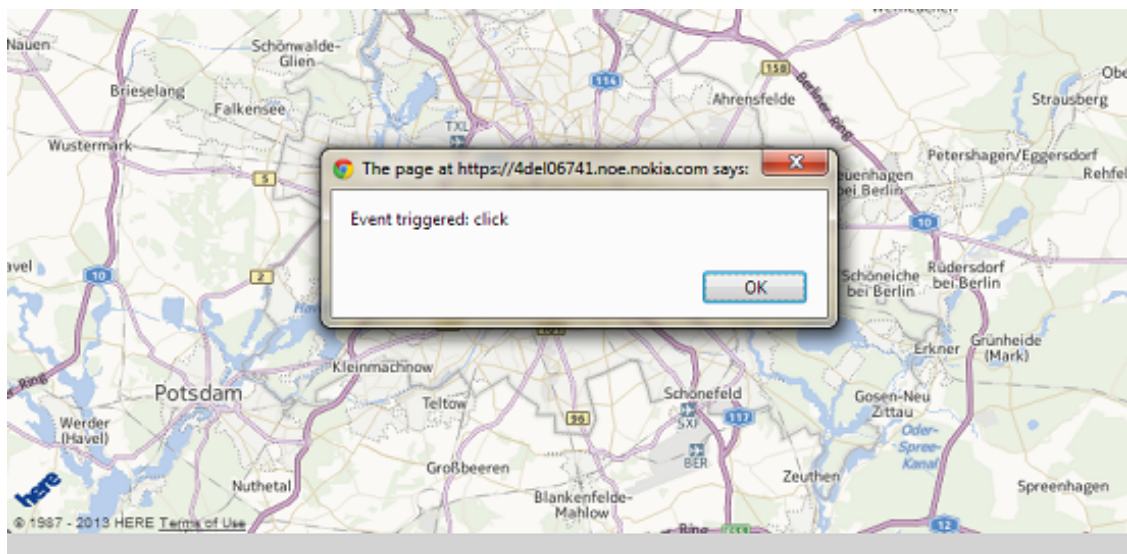

The image below shows the initial view an HTML page including the above code.

Figure 14: A map with a clickable area for the Simple Example for Page events



The picture that follows shows the same page after the user has clicked on the rectangle under the map. The alert reflects the fact that the click event has been processed.

Figure 15: Feedback indicating that the Page event has been processed



Observers and Mutable Objects

Tracking state changes is a common use case and the Maps API provides an implementation of the *Observer* pattern to address it. It allows you to observe:

- changes of state – for objects derived from `Object`
- actions – for instances of `OList`

In both cases, you need to call the method `addObserver()` on the object in which you are interested, passing to it a callback function to be invoked when an observable change to the object occurs. In addition, you need to indicate the property or properties to observe on an instance of `Object`. By contrast, an `OList` observer (callback) is invoked on all and any actions that affect the state of the object, so when adding an `OList` observer, there is no option to specify the actions or properties in which you are interested.

The example below shows how to add an observer on an instance of `map.Display` represented by a variable named `map`. `map.Display` extends `Object`, therefore the callback receives information about the object's state changes, and specifically about the property named "state" – the callback receives it via the argument `value`.

```
var zoomObserver = function (obj, key, newValue, oldValue) {  
  if (newValue > 14) {  
    map.set("zoomLevel", oldValue);  
  }  
};  
  
map.addObserver("zoomLevel", zoomObserver);
```

In this case, the observer is invoked whenever the `Display` property "zoomLevel" changes. It checks if the new value of the property is higher than 14, and if it is, it makes sure "zoomLevel" reverts to the last known value before the change. In short: it is used to ensure that "zoomLevel" can never be higher than 14.

Mutable Objects

The classes defined in the Maps API distinguish between two types of objects:

- `immutable` – objects whose state should never change once they have been created
- `mutable` – objects that are designed to hold data which may be altered

All mutable objects are derived either from `OList` or `Object` and therefore offer the *Observer* pattern described in (see [Observers and Mutable Objects](#) on page 49).

here

An example of an object that is designed to be `immutable` is an instance of the class `Coordinate` from the namespace `geo`. The class is not designed to support state changes and therefore it does not offer the *Observer* pattern. Although technically JavaScript permits you to alter the internal state of an `immutable` object, bear in mind that this could lead to problems, as there is no mechanism to propagate the state change information to other objects that may depend on it.

Please check the inheritance chain in the documentation to be sure if a class is observable or not.

Multi-language Support

The HERE Maps API for JavaScript supports localization in the following languages:

Table 9: Localization Languages in HERE Maps API for JavaScript

Locale ID	Description	Comment
en-GB	English as spoken in Great Britain	
en-US	English as spoken in the USA	default language
de-DE	German as spoken in Germany	
fr-FR	French as spoken in France	
es-ES	Spanish as spoken in Spain	
it-IT	Italian as spoken in Italy	
ru-RU	Russian as spoken in Russia	
zh-CN	Simplified Chinese	

Please note that the list of supported languages is under constant revision and HERE reserves the right to alter the list without notice.

By default, the language of the maps displayed using this API is defined by the browser's language preferences. You can override this behavior to meet specific requirements by calling the method `set()` on the static class `Settings` before you initialize the map (otherwise the attempt to set the language has no effect). The first argument to `set()` is the name of the property you wish to change, in this case `"defaultLanguage"`. The second argument is the value of the property. The following example sets the default language to German spoken in Germany by calling the method `set()` on the class `Settings`:

```
nokia.Settings.set("defaultLanguage", "de-DE");
var map = new nokia.maps.map.Display(
```

```
document.getElementById("mapContainer"),
{
  components: [
    new nokia.maps.map.component.ZoomBar(),
    new nokia.maps.map.component.TypeSelector()],
  zoomLevel: 2,
  center: [40.83, 28.98]
};
```

Note that if you set the locale to an unsupported language, the API automatically attempts to use the closest supported alternative or the default language for the API ("en-US"). For example, if you set "language" to "fr-BE" (French as spoken in Belgium), the map text is automatically displayed in French as spoken in France ("fr-FR").

Chapter

4

API reference

Topics:

- [Namespace:](#)

The following pages offer a complete detailed reference to the namespaces, classes and methods in the Maps API for JavaScript.

here

Namespace:

Namespace Summary

Class: Features

This class is a member of [nokia](#).

Class Summary

This class provides facilities that allow you to register and load individual functional components (features) of the API.

[For full details, see [nokia.Features](#)]

Table 10: Method Summary

Methods
static add (featureName, featureImplName, loadPath, detector, dependencies, overrides, charset) This method adds a new feature implementation definition to the feature registry.
static get (featureName, featureImplName) This method retrieves a feature implementation definition object for the feature specified by the caller.
static getFeatureMap () This method retrieves a map object containing the names of available implementations for each registered feature.
static getLoadedMap () This method retrieves a map of the names of fully loaded implementations for each registered feature.
static isLoading (featureName, featureImplName) : {Boolean} This method checks whether a certain feature implementation has been successfully loaded.
static load (requested, onSuccess, onError, doc=document, sync) This method loads a set of features specified by the caller.

here

Class Description

This class provides facilities that allow you to register and load individual functional components (features) of the API. It automatically resolves dependencies between features and provides mechanisms to detect the best feature implementation for the current environment. A feature may exist in different implementations for different target environments. A feature implementation is a set of classes that provide certain functionality and can be loaded to enrich the functionality set accessible through the API. The features that can be loaded individually with the help of the methods on this class include such components as "routing" and "search".

By default, the API loads package named "all" (see section "Packages and Detection" in User Guide) which contains features in optimal implementation for the environment in which it runs. However, the loading of packages may be deferred at page-load time. This may reduce the initial page load time by limiting the amount of API code that is transferred and executed before the page contents are fully displayed. To do that loader file `jsl.js` must be called with `blank=true` GET parameter (ex. `<script src="http://js.api.here.com/se/2.5.4/jsl.js?blank=true" type="text/javascript"></script>`).

Features is a static object that cannot be instantiated, therefore to call one of its methods, you must precede the method name with the namespace and the name of the class, using the dot notation. For example, in case of deferred loading you can call the `load()` method as follows:

```
var featureMap = nokia.Features.getFeaturesFromMatrix(["maps"]); //that will
    return all features that are included in "maps" package
nokia.Features.load(featureMap, successCallback, errorCallback, document,
    false); //loads asynchronously all necessary files and
    //invokes successCallback where map instantiation etc. can take place
```

Method Details

`static add(featureName, featureImplName, loadPath, detector, dependencies, overrides, charset)`

This method adds a new feature implementation definition to the feature registry. If the feature does not yet exist in the registry, an entry for it is created, including the feature name and the implementation and the implementation object.

Parameters:

featureName: {string}

A canonical name of the feature which the implementation provides

featureImplName: {string}

A canonical the name of the implementation

here

loadPath:	{string}	The load path of the implementation file
detector:	{function ()} [optional]	The detector function that determines whether this feature can be loaded in the current environment; if a detector is not provided, a default function that always returns true is used
dependencies:	{Array} [optional, default: null]	An optional array of the names of the features on which the named implementation depends
overrides:	{Array} [optional, default: null]	An optional array of implementation names (within the same feature) this implementation overrides.
charset:	{string} [optional, default: 'utf-8']	An optional identifier of the character set for the script

static **get**(featureName, featureImplName)

This method retrieves a feature implementation definition object for the feature specified by the caller.

Parameters:

featureName:	{string}	The name of the feature to be retrieved
featureImplName:	{string} [optional, default: 'auto']	Either the name of the specific implementation or 'auto', which allows the method automatically to detect the optimal implementation for the current environment

Returns:

here

A feature implementation object, or `null` if the feature is not supported by the current environment

`static` **getFeatureMap()**

This method retrieves a map object containing the names of available implementations for each registered feature. The map has the following structure:

```
{
  featureA: [featureImplNameA1, featureImplNameA2, ...],
  featureB: [featureImplNameB1, ...]
  ...
}
```

Returns:

A map object listing registered feature implementations

`static` **getLoadedMap()**

This method retrieves a map of the names of fully loaded implementations for each registered feature. The returned map has the following structure:

```
{
  featureA: [featureImplNameA1, featureImplNameA2, ...],
  featureB: [featureImplNameB1, ...]
  ...
}
```

Returns:

A map object listing loaded implementations of registered features

`static` **isLoaded**(featureName, featureImplName): {Boolean}

This method checks whether a certain feature implementation has been successfully loaded. The method returns `true` if the queried implementation has been successfully loaded (i.e. transmitted AND evaluated). The method throws an exception when either the feature or the implementation is unknown.

Parameters:

featureName: {Object}

here

The name of the feature to check

featureImplName: {Object}

- the name of the implementation of the feature to check

Returns:

{Boolean} true if the feature has been fully loaded, false otherwise

static **load**(requested, onSuccess, onError, doc=document, sync)

This method loads a set of features specified by the caller. The caller can provide optional callbacks to be invoked on success and on error and also a target document to which the respective script tags should be appended. The caller specifies the features to load in a hash object, where the keys represent the feature names and the values provide the implementation parameters, for example:

```
nokia.Features.load({
  "map": "auto",
  "search": "auto"
});
```

Parameters:

requested: {Object}

A hash containing the names of the features to load (as keys) and feature implementation parameters (as values)

onSuccess: {function ()}

A function to be called on success

onError: {function (e)}

A function to be called on error

doc=document: {Document}

The host document

sync: {Boolean}

here

A Boolean indicating whether synchronous loading via `document.write` is to be enforced (`true`) or not (`false`)

Class: Settings

This class is a member of [nokia](#).

Class Summary

This is an observable static class that encapsulates API settings.

[For full details, see [nokia.Settings](#)]

Table 11: Property Summary

Properties
static app_code: {String} This property holds a string token which authenticates the requested application or Web page.
static app_id: {String} This property holds an id which authenticates the requested application or Web page.
~~static appId: {String}~~ This property holds an id which authenticates the requested application or Web page.
~~static authenticationToken: {String}~~ This property holds a string token which authenticates the requested application or Web page.
static defaultLanguage: {String} This property holds the default language used by all service managers.
static secureConnection: {String} This property holds a value indicating whether the API must use HTTPS connections to communicate with a back-end service.
static serviceMode: {String} This property defines whether the API should send service requests to production URLs or URLs for demo purposes, customer integration testing, etc.

Table 12: Method Summary

Methods
static addObserver (name, callback, context) This method registers an observer for the property named by the caller.
static removeObserver (name, callback, context)

here

Methods

This method unregisters a previously registered observer for the property named by the caller.

static **set** (name, value, force)

This method sets properties, using the property names and values supplied by the caller.

Class Description

This is an observable static class that encapsulates API settings. It stores API-wide data such as defaultLanguage, appId, and authenticationToken. Properties of the class must be set via the method set(), rather than directly.

Example:

```
nokia.Settings.set("defaultLanguage", "en-GB");
nokia.Settings.set("app_id", "YOUR_APPID");
nokia.Settings.set("app_code", "YOUR_APPCODE");
```

Property Details

static **app_code**: {String}

This property holds a string token which authenticates the requested application or Web page. The third-party offering comes with an in-built token, limited to a certain amount of requests per day. The property must be set via the method set().

static **app_id**: {String}

This property holds an id which authenticates the requested application or Web page. The third-party offering comes with an inbuilt id, limited to a certain amount of requests per day. The property must be set via the method set().

~~static **appId**: {String}~~

This property holds an id which authenticates the requested application or Web page. The third-party offering comes with an inbuilt id, limited to a certain amount of requests per day. The property must be set via the method set().

Deprecated: This property is deprecated, please use app_id.

~~static **authenticationToken**: {String}~~

here

This property holds a string token which authenticates the requested application or Web page. The third-party offering comes with an in-built token, limited to a certain amount of requests per day. The property must be set via the method `set()`.

Deprecated: This property is deprecated, please use `app_code`.

static `defaultLanguage`: {String}

This property holds the default language used by all service managers. The value must be a string combining an ISO639-1 language code and the appropriate ISO3166-1 alpha-2 country code, for example, "en-US". The property must be set via the method `set()`.

static `secureConnection`: {String}

This property holds a value indicating whether the API must use HTTPS connections to communicate with a back-end service. To enable HTTPS communication, set the value to "force".

Default Value: undefined

static `serviceMode`: {String}

This property defines whether the API should send service requests to production URLs or URLs for demo purposes, customer integration testing, etc. The following values are possible:

- `undefined` - Indicates that the API will send requests to default production URLs. This is the default value.
- `"cit"` - Indicates that the API will send requests to URLs for customer integration testing

Default Value: undefined

Method Details

static `addObserver`(name, callback, context)

This method registers an observer for the property named by the caller.

Parameters:

name: {String}

The name of the property to observe

here

callback: {Function}

The function to be called if the observed property is modified; the function must be able to receive the following arguments:

- (Variant) value - the new value that the property should be set to

context: {Object} [optional]

The context in which the given function should be called (default null)

static **removeObserver**(name, callback, context)

This method unregisters a previously registered observer for the property named by the caller.

Parameters:

name: {Object}

The name of the property that is no longer to be observed (the observer for this property is to be removed)

callback: {Object}

The observer function to remove

context: {Object} [optional]

The context objects that was received along with the observer function when the observer was registered or none if the observer was registered without a context

static **set**(name, value, force)

This method sets properties, using the property names and values supplied by the caller.

Parameters:

name: {String}

The name of the property to be set

value: {Variant}

here

The value that should be applied to the property

force: {Boolean}

this will force notifying observers even if value is the same

Namespace: maps

This namespace is a member of [nokia](#).

Namespace Summary

This namespace contains the implementation classes of Nokia's Maps API.

Namespace Description

This namespace contains the implementation classes of Nokia's Maps API.

Namespace: advrouting

This namespace is a member of [nokia.maps](#).

Namespace Summary

This namespace defines classes that are required for extended routing use cases.

Namespace Description

This namespace defines classes that are required for extended routing use cases.

Interface: CalculateIsolineRequest

This interface is a member of [nokia.maps.advrouting](#).

Interface Summary

This interface defines an isoline request.

[For full details, see [nokia.maps.advrouting.CalculateIsolineRequest](#)]

here

Table 13: Property Summary

Properties
<code>avoidArea</code>: {<i>nokia.maps.geo.BoundingBox</i>[]} This property holds an array of <i>BoundingBox</i> objects that define areas which the routes must not cross.
<code>avoidLinks</code>: {String[]} This property holds a list of links the route must avoid.
<code>departure</code>: {Number} This property specifies the start time to be used in route calculations, allowing the routing engine to take into consideration time-dependent traffic patterns and incidents when calculating the route.
<code>distances</code>: {Number[]} Total travel distance, must be less than 10 km.
<code>metaInfo</code>: {<i>nokia.maps.advrouting.CalculateIsolineRequestMetaInfo</i>} This property holds a reference to an optional <i>nokia.maps.advrouting.CalculateIsolineRequestMetaInfo</i> object.
<code>mode</code>: {<i>nokia.maps.advrouting.Mode</i>} This property specifies the transport mode to be used when calculating the routes.
<code>start</code>: {<i>nokia.maps.advrouting.WaypointParameter</i>} This property defines the starting point for the isoline route calculations.
<code>travelTimes</code>: {Number} Total travel time, must be less than 10 minutes.
<code>truckProfile</code>: {<i>nokia.maps.advrouting.TruckProfile</i>} This property holds a reference to an optional <i>nokia.maps.advrouting.TruckProfile</i> object.

Interface Description

In the routing context, an isoline is a polyline that connects the end points of all routes leaving from a common point of origin and have the same journey length either in terms of the distance covered or travel time. The interface defines the request parameters to retrieve multiple isolines.

Property Details

`avoidArea`: {[*nokia.maps.geo.BoundingBox*](#)[]}

This property holds an array of [*BoundingBox*](#) objects that define areas which the routes must not cross.

`avoidLinks`: {String[]}

here

This property holds a list of links the route must avoid.

departure: {Number}

This property specifies the start time to be used in route calculations, allowing the routing engine to take into consideration time-dependent traffic patterns and incidents when calculating the route.

distances: {Number[]}

Total travel distance, must be less than 10 km. Specify either distance or time, not both.

metaInfo: {[*nokia.maps.advrouting.CalculateIsolineRequestMetaInfo*](#)}

This property holds a reference to an optional [*nokia.maps.advrouting.CalculateIsolineRequestMetaInfo*](#) object.

mode: {[*nokia.maps.advrouting.Mode*](#)}

This property specifies the transport mode to be used when calculating the routes. The transport mode has an impact on the speed information provided at link level.

start: {[*nokia.maps.advrouting.WaypointParameter*](#)}

This property defines the starting point for the isoline route calculations.

travelTimes: {Number}

Total travel time, must be less than 10 minutes. Specify either distance or time, not both.

truckProfile: {[*nokia.maps.advrouting.TruckProfile*](#)}

This property holds a reference to an optional [*nokia.maps.advrouting.TruckProfile*](#) object.

Interface: CalculateIsolineRequestMetaInfo

This interface is a member of [*nokia.maps.advrouting*](#).

here

Interface Summary

This interface defines meta information for isoline routing calculation requests.

[For full details, see [nokia.maps.advrouting.CalculateIsolineRequestMetaInfo](#)]

Table 14: Property Summary

Properties
<code>requestId</code> : {String} This property holds an arbitrary value to be echoed in the response to the route calculation request.
<code>verboseMode</code> : {Number} This property holds a numeric value indicating the required verbosity level in the response structure.

Interface Description

This interface describes the structure of an object that can be passed as meta information to an instance of [nokia.maps.advrouting.CalculateIsolineRequest](#). The meta information represents additional information that is not specific to route retrieval.

Property Details

`requestId`: {String}

This property holds an arbitrary value to be echoed in the response to the route calculation request. The value can be used to trace back request.

`verboseMode`: {Number}

This property holds a numeric value indicating the required verbosity level in the response structure. The permitted values are in the range between 0 and 5.

Interface: CalculateIsolineResponse

This interface is a member of [nokia.maps.advrouting](#).

Interface Summary

This interface defines a response to a request for an isoline route calculation represented by [nokia.maps.advrouting.CalculateIsolineRequest](#).

[For full details, see [nokia.maps.advrouting.CalculateIsolineResponse](#)]

here

Table 15: Property Summary

Properties
center: {nokia.maps.geo.Coordinate} This property holds the coordinates of the start point defined in the isoline calculation request and is the point of origin of all the routes included in the isolines resulting from the calculation.
isolines: {nokia.maps.geo.Shape[]} This property holds a list of lines that connect the end points of routes with the specified length/duration from the center.
metaInfo: {nokia.maps.advrouting.RouteResponseMetaInfo} This property holds a reference to an optional nokia.maps.advrouting.RouteResponseMetaInfo object.

Interface Description

This interface defines a response to a request for an isoline route calculation represented by [nokia.maps.advrouting.CalculateIsolineRequest](#).

Property Details

center: {[nokia.maps.geo.Coordinate](#)}

This property holds the coordinates of the start point defined in the isoline calculation request and is the point of origin of all the routes included in the isolines resulting from the calculation. request.

isolines: {[nokia.maps.geo.Shape](#)[]}

This property holds a list of lines that connect the end points of routes with the specified length/duration from the center. Each length/duration entry in the request corresponds to one isoline in the response (with maintained order).

metaInfo: {[nokia.maps.advrouting.RouteResponseMetaInfo](#)}

This property holds a reference to an optional [nokia.maps.advrouting.RouteResponseMetaInfo](#) object.

Class: CalculateRouteRequest

This class is a member of [nokia.maps.advrouting](#).

Extends: [nokia.maps.util.OObject](#)

here

Class Summary

This class represents a request for a route calculation.

[For full details, see [nokia.maps.advrouting.CalculateRouteRequest](#)]

Table 16: Property Summary

Properties
avoidArea: {nokia.maps.geo.BoundingBox[]} This property holds a list of bounding boxes that define areas which the route must not cross.
avoidLinks: {String[]} This property holds a list of links that the route may not cross.
departure: {Number} This property holds a value indicating the expected departure time.
metaInfo: {nokia.maps.advrouting.CalculateRouteRequestMetaInfo} This property holds a reference to an optional nokia.maps.advrouting.CalculateRouteRequestMetaInfo object.
metricSystem: {String} This property holds value indicating the metric system for returned data.
modes: {nokia.maps.advrouting.Mode[]} This property specifies the transport mode to be used when calculating the routes.
representationOptions: {nokia.maps.advrouting.RouteRepresentationOptions} This property holds a reference to an optional nokia.maps.advrouting.RouteRepresentationOptions object.
truckProfile: {nokia.maps.advrouting.TruckProfile} This property holds a reference to an optional nokia.maps.advrouting.TruckProfile object.
waypoints: {nokia.maps.advrouting.WaypointParameter[]} This property holds an array of waypoints through which the route should pass (in the given order).

Table 17: Method Summary

Methods
setDestination (destinationWaypoint) This is a convenience method that adds the destination point to route.
setStart (startWaypoint) This is a convenience method that adds the start point to the route.

Directly Inherited Methods

here

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This class represents a request for a route calculation.

Constructor Details

nokia.maps.advrouting.CalculateRouteRequest()

This constructor creates a new CalculateRouteRequest object.

Property Details

avoidArea: {[nokia.maps.geo.BoundingBox](#)[]}

This property holds a list of bounding boxes that define areas which the route must not cross.

avoidLinks: {String[]}

This property holds a list of links that the route may not cross.

departure: {Number}

This property holds a value indicating the expected departure time. The routing engine takes into consideration time-dependent traffic patterns and incidents when calculating the route. The route request can specify either the arrival or the departure time.

metaInfo: {[nokia.maps.advrouting.CalculateRouteRequestMetaInfo](#)}

This property holds a reference to an optional [nokia.maps.advrouting.CalculateRouteRequestMetaInfo](#) object.

metricSystem: {String}

This property holds value indicating the metric system for returned data. It can be specified by two values 'metric' or 'imperial'.

here

modes: {[nokia.maps.advrouting.Mode](#)[]}

This property specifies the transport mode to be used when calculating the routes. The transport mode has an impact on the speed information provided at link level.

representationOptions: {[nokia.maps.advrouting.RouteRepresentationOptions](#)}

This property holds a reference to an optional [nokia.maps.advrouting.RouteRepresentationOptions](#) object.

truckProfile: {[nokia.maps.advrouting.TruckProfile](#)}

This property holds a reference to an optional [nokia.maps.advrouting.TruckProfile](#) object.

waypoints: {[nokia.maps.advrouting.WaypointParameter](#)[]}

This property holds an array of waypoints through which the route should pass (in the given order). The first element marks the start, the last the end point of the route. The waypoints in between are interpreted as via points.

Method Details

setDestination(destinationWaypoint)

This is a convenience method that adds the destination point to route.

Parameters:

destinationWaypoint: {[nokia.maps.advrouting.WaypointParameter](#)}

An object defining a waypoint that is to be the destination point of the route

setStart(startWaypoint)

This is a convenience method that adds the start point to the route.

Parameters:

startWaypoint: {[nokia.maps.advrouting.WaypointParameter](#)}

An object defining a waypoint that is to be the start point of the route

here

Interface: CalculateRouteRequestMetaInfo

This interface is a member of [nokia.maps.advrouting](#).

Interface Summary

This interface defines meta information that can be associated with a route calculation request.

[For full details, see [nokia.maps.advrouting.CalculateRouteRequestMetaInfo](#)]

Table 18: Property Summary

Properties
requestId : {String} This property holds an arbitrary value to be echoed in the response to the route calculation request.

Interface Description

This interface describes the structure of an object which can be passed as meta information (or additional information) to an instance of [nokia.maps.advrouting.CalculateRouteRequest](#).

Property Details

requestId: {String}

This property holds an arbitrary value to be echoed in the response to the route calculation request. The value can be used to trace requests.

Interface: CalculateRouteResponse

This interface is a member of [nokia.maps.advrouting](#).

Interface Summary

This interface defines a data structure for the responses to route calculation requests.

[For full details, see [nokia.maps.advrouting.CalculateRouteResponse](#)]

Table 19: Property Summary

Properties
metaInfo : { nokia.maps.advrouting.RouteResponseMetaInfo } This property holds a mandatory reference to an instance of nokia.maps.advrouting.RouteResponseMetaInfo .

here

Properties

~~**route:** {[nokia.maps.advrouting.Route](#)[]}~~

This property holds an array of [nokia.maps.advrouting.Route](#) objects.

routes: {[nokia.maps.advrouting.Route](#)[]}

This property holds an array of [nokia.maps.advrouting.Route](#) objects.

Interface Description

This interface describes the structure of the object that represents a response to a request of type [nokia.maps.advrouting.CalculateRouteRequest](#).

Property Details

metaInfo: {[nokia.maps.advrouting.RouteResponseMetaInfo](#)}

This property holds a mandatory reference to an instance of [nokia.maps.advrouting.RouteResponseMetaInfo](#).

~~**route:** {[nokia.maps.advrouting.Route](#)[]}~~

This property holds an array of [nokia.maps.advrouting.Route](#) objects. Note that the array can be empty.

Deprecated: since 2.5.4

routes: {[nokia.maps.advrouting.Route](#)[]}

This property holds an array of [nokia.maps.advrouting.Route](#) objects. Note that the array can be empty.

Interface: GeoWaypointParameter

This interface is a member of [nokia.maps.advrouting](#).

Interface Summary

This interface represents a waypoint based on an instance of [nokia.maps.geo.GeoCoordinate](#).

[For full details, see [nokia.maps.advrouting.GeoWaypointParameter](#)]

here

Table 20: Property Summary

Properties
<i>position</i> : { <i>nokia.maps.geo.Coordinate</i> } This property holds an object containing the geographic coordinates as a center around which to locate route links.
<i>transitRadius</i> : {Number} This property holds a value indicating the radius within which to locate/select matching links.

Interface Description

This interface represents a waypoint based on an instance of *nokia.maps.geo.GeoCoordinate*. In a common use case, a use can specify a waypoint by clicking on the map.

Property Details

position: {*nokia.maps.geo.Coordinate*}

This property holds an object containing the geographic coordinates as a center around which to locate route links.

transitRadius: {Number}

This property holds a value indicating the radius within which to locate/select matching links.

Interface: GetRouteRequest

This interface is a member of *nokia.maps.advrouting*.

Interface Summary

This interface defines a data structure that contains the details of route calculation request to the GetRoute service.

[For full details, see *nokia.maps.advrouting.GetRouteRequest*]

Table 21: Property Summary

Properties
<i>avoidArea</i> : { <i>nokia.maps.geo.BoundingBox</i> []} This property holds an array of objects that represent areas which the route must not cross.
<i>avoidLinks</i> : {String[]}

here

Properties

This property holds an array of id's of route link the route must avoid.

currentPosition: {*nokia.maps.advrouting.WaypointParameter*}

This property holds the current position as an instance of *nokia.maps.advrouting.WaypointParameter*.

departure: {Number}

This property holds the time when the journey is expected to start.

metaInfo: {*nokia.maps.advrouting.GetRouteRequestMetaInfo*}

This property holds a reference to an instance of *nokia.maps.advrouting.GetRouteRequestMetaInfo*.

metricSystem: {String}

The default is the metric system associated with the language settings.

mode: {*nokia.maps.advrouting.Mode*[]}

This property holds an array of objects specifying modes that has been used when calculating the route.

representationOptions: {*nokia.maps.advrouting.RouteRepresentationOptions*}

This property holds a reference to an instance of *nokia.maps.advrouting.RouteRepresentationOptions*.

routeId: {String}

This property holds the id of the route object that represents a previously calculated route.

truckProfile: {*nokia.maps.advrouting.TruckProfile*}

This property holds a reference to an instance of *nokia.maps.advrouting.TruckProfile*.

Interface Description

This interface defines a data structure that contains the details of route calculation request to the GetRoute service. The main input parameter for the service is routeId returned by a previous route calculation and represents the entire route. In a corridor routing scenario, additional information besides routeId is needed to reconstruct the corridor around the base type.

Property Details

avoidArea: {*nokia.maps.geo.BoundingBox*[]}

This property holds an array of objects that represent areas which the route must not cross.

avoidLinks: {String[]}

This property holds an array of id's of route link the route must avoid.

here

currentPosition: {[nokia.maps.advrouting.WaypointParameter](#)}

This property holds the current position as an instance of [nokia.maps.advrouting.WaypointParameter](#). If the current position is provided, the request response includes information about the travel progress with updated remaining travel times.

departure: {Number}

This property holds the time when the journey is expected to start. The routing engine considers time-dependent traffic patterns and incidents when calculating the route.

metaInfo: {[nokia.maps.advrouting.GetRouteRequestMetaInfo](#)}

This property holds a reference to an instance of [nokia.maps.advrouting.GetRouteRequestMetaInfo](#). The use of this property is optional.

metricSystem: {String}

The default is the metric system associated with the language settings..

- imperial
- metric

mode: {[nokia.maps.advrouting.Mode](#)[]}

This property holds an array of objects specifying modes that has been used when calculating the route. The transport mode has an impact on the speed information provided at link level.

representationOptions: {[nokia.maps.advrouting.RouteRepresentationOptions](#)}

This property holds a reference to an instance of [nokia.maps.advrouting.RouteRepresentationOptions](#). The use of this property is optional.

routeId: {String}

This property holds the id of the route object that represents a previously calculated route. It is required to reconstruct exactly the same route.

here

truckProfile: {[nokia.maps.advrouting.TruckProfile](#)}

This property holds a reference to an instance of [nokia.maps.advrouting.TruckProfile](#). The use of this property is optional.

Interface: GetRouteRequestMetaInfo

This interface is a member of [nokia.maps.advrouting](#).

Interface Summary

This interface defines a set of route request parameters which are not specific to route retrieval.

[For full details, see [nokia.maps.advrouting.GetRouteRequestMetaInfo](#)]

Table 22: Property Summary

Properties
RequestId: {String} This property holds an arbitrary value to be mirrored in the response structure.

Interface Description

This interface defines a set of route request parameters which are not specific to route retrieval.

Property Details

RequestId: {String}

This property holds an arbitrary value to be mirrored in the response structure. The value can be used to trace back requests.

Interface: GetRouteResponse

This interface is a member of [nokia.maps.advrouting](#).

Interface Summary

This interface defines a response to a route calculation request.

[For full details, see [nokia.maps.advrouting.GetRouteResponse](#)]

here

Table 23: Property Summary

Properties
<code>metaInfo</code> : { <code>nokia.maps.advrouting.RouteResponseMetaInfo</code> } This property holds a mandatory reference to an instance of <code>RouteResponseMetaInfo</code> .
<code>progress</code> : { <code>nokia.maps.advrouting.TravelProgress</code> } This property holds a reference to an instance of <code>TravelProgress</code> (optional).
<code>route</code> : { <code>nokia.maps.advrouting.Route</code> } This property holds a reference to an <code>Route</code> object (optional).

Interface Description

This interface defines a response to a route calculation request.

Property Details

`metaInfo`: {[`nokia.maps.advrouting.RouteResponseMetaInfo`](#)}

This property holds a mandatory reference to an instance of `RouteResponseMetaInfo`.

`progress`: {[`nokia.maps.advrouting.TravelProgress`](#)}

This property holds a reference to an instance of `TravelProgress` (optional).

`route`: {[`nokia.maps.advrouting.Route`](#)}

This property holds a reference to an `Route` object (optional).

Interface: LinkPosition

This interface is a member of [`nokia.maps.advrouting`](#).

Interface Summary

This interface defines an accurate position of a route link.

[For full details, see [`nokia.maps.advrouting.LinkPosition`](#)]

here

Table 24: Property Summary

Properties
linkId: {String} This property holds the id of the given link position.
sideOfStreet: {String} This property indicates whether the waypoint is on the left or right side of the link (if heading from the reference node to the non-reference node).
spot: {Number} This property holds the relative position of a link location defined as the fractional distance from the link's reference-node to the non-reference node, i.e.

Interface Description

This interface defines an accurate position of a route link.

Property Details

linkId: {String}

This property holds the id of the given link position.

sideOfStreet: {String}

This property indicates whether the waypoint is on the left or right side of the link (if heading from the reference node to the non-reference node). The possible values are:

- "left"
- "right"
- "neither"

spot: {Number}

This property holds the relative position of a link location defined as the fractional distance from the link's reference-node to the non-reference node, i.e. the value range is between 0 and 1. This attribute is only relevant if a link is referenced.

Class: Manager

This class is a member of *nokia.maps.advrouting*.

here

Extends: [nokia.maps.routing.Manager](#)

Class Summary

This class represents the client-side interface to routing functionality.

[For full details, see [nokia.maps.advrouting.Manager](#)]

Table 25: Property Summary

Properties
calculateIsolineResponse : { nokia.maps.advrouting.CalculateIsolineResponse }
This property holds an instance of a response to a request for an isoline route calculation.
calculateRouteResponse : { nokia.maps.advrouting.CalculateRouteResponse }
This property holds an instance of a response to a request for a route calculation.
getRouteResponse : { nokia.maps.advrouting.GetRouteResponse }
This property holds a reference to a routing request response.

Directly Inherited Properties

Inherited from class [nokia.maps.routing.Manager](#) :

[metricSystem](#), [state](#)

Table 26: Method Summary

Methods
calculateIsoline (request)
This method submits a request for an isoline route calculation.
calculateRoute (request)
This method submits a request for a route calculation.
getErrorCause () : { nokia.maps.routing.ServiceError }
This method retrieves an object describing the cause of an error that occurred when a request was being processed.
getIsolines () : { nokia.maps.geo.Shape []}
This method retrieves the isoline routes calculated by the server.
getRoute (request)
This method requests existing routes from the server.

Directly Inherited Methods

Inherited from class [nokia.maps.routing.Manager](#) :

here

calculateRoute, clear, destroy, getErrorCause, getRoutes

Inherited from class *nokia.maps.util.OObject* :

addObserver, get, remove, removeObserver, set

Class Description

This class represents the client-side interface to routing functionality. We assume that only one routing response is kept at a time. If a new request is triggered the old response is replaced.

Constructor Details

nokia.maps.advrouting.Manager()

This constructor creates a new empty Manager object.

Property Details

calculateIsolineResponse: {*nokia.maps.advrouting.CalculateIsolineResponse*}

This property holds an instance of a response to a request for an isoline route calculation. Property is populated after successful `calculateIsoline` call.

calculateRouteResponse: {*nokia.maps.advrouting.CalculateRouteResponse*}

This property holds an instance of a response to a request for a route calculation. Property is populated after successful `calculateRoute` call.

getRouteResponse: {*nokia.maps.advrouting.GetRouteResponse*}

This property holds a reference to a routing request response. Property is populated after successful `getRoute` call.

Method Details

calculateIsoline(request)

This method submits a request for an isoline route calculation.

Parameters:

request: {*nokia.maps.advrouting.CalculateIsolineRequest*}

here

An object that specifies the details of the request.

calculateRoute(request)

This method submits a request for a route calculation.

Parameters:

request: {*nokia.maps.advrouting.CalculateRouteRequest*}

A request object that specifies the details of the request.

getErrorCause(): {*nokia.maps.routing.ServiceError*}

This method retrieves an object describing the cause of an error that occurred when a request was being processed.

Returns:

{*nokia.maps.routing.ServiceError*}

The error object containing the error type}

getIsolines(): {*nokia.maps.geo.Shape*[]}

This method retrieves the isoline routes calculated by the server.

Returns:

{*nokia.maps.geo.Shape*[]}

getRoute(request)

This method requests existing routes from the server.

Parameters:

request: {*nokia.maps.advrouting.GetRouteRequest*}

An object that specifies the details of the request.

here

Interface: Maneuver

This interface is a member of [nokia.maps.advrouting](#).

Interface Summary

This interface defines a maneuver, which describes actions the person following the route must carry out to enter the next link or segment of the route.

[For full details, see [nokia.maps.advrouting.Maneuver](#)]

Table 27: Property Summary

Properties
boundingBox : { nokia.maps.geo.BoundingBox } Coordinates defining the bounding box of the entire maneuver.
equipment : {String[]} This property holds an array of identifiers of equipment and features available at the location of the maneuver.
firstPoint : {Number} Index into the global geometry array, pointing to the first point of the shape subsegment associated with this Maneuver.
instruction : {String} This property holds the maneuver instructions, for example "Turn left onto Minna St.
lastPoint : {Number} Index into the global geometry array, pointing to the last point of the shape subsegment associated with this Maneuver.
length : {Number} This property holds the length of the segment following the given maneuver until the next maneuver.
nextManeuver : { nokia.maps.advrouting.Maneuver } This property holds a reference to an object representing the next maneuver on the route.
note : { nokia.maps.advrouting.RouteNote } This property holds additional information about the route segment, following the maneuver, for example "sharp curve ahead", "accessing toll road", etc.
position : { nokia.maps.geo.Coordinate } This property holds the geographic coordinates of the location, where the maneuver starts.
shape : { nokia.maps.geo.Shape } This property holds an object representing the shape of the route segment following the given maneuver as a polyline.
time : {Number} This property holds a value indicating the time when the given maneuver is to occur.

here

Properties

toLink: {*nokia.maps.advrouting.RouteLink*}

This property holds a reference to an object representing the next link on the route.

travelTime: {Number}

This property holds the time needed to travel the length of the segment following the given maneuver until the next maneuver.

Interface Description

This interface defines a maneuver, which describes actions the person following the route must carry out to enter the next link or segment of the route.

Property Details

boundingBox: {*nokia.maps.geo.BoundingBox*}

Coordinates defining the bounding box of the entire maneuver.

equipment: {String[]}

This property holds an array of identifiers of equipment and features available at the location of the maneuver. The possible value are:

- "stairs"
- "elevator"
- "escalator"
- "barrierFree"
- "shelter"
- "restroom"
- "shopping"
- "restaurant"
- "baggageRoom"
- "bikeDepot"
- "bikeAccepted"
- "parking"
- "unknown"

here

firstPoint: {Number}

Index into the global geometry array, pointing to the first point of the shape subsegment associated with this Maneuver. Must be followed by `lastPoint`.

instruction: {String}

This property holds the maneuver instructions, for example "Turn left onto Minna St."

lastPoint: {Number}

Index into the global geometry array, pointing to the last point of the shape subsegment associated with this Maneuver. Must be preceded by `firstPoint`

length: {Number}

This property holds the length of the segment following the given maneuver until the next maneuver.

nextManeuver: {*nokia.maps.advrouting.Maneuver*}

This property holds a reference to an object representing the next maneuver on the route.

note: {*nokia.maps.advrouting.RouteNote*}

This property holds additional information about the route segment, following the maneuver, for example "sharp curve ahead", "accessing toll road", etc.

position: {*nokia.maps.geo.Coordinate*}

This property holds the geographic coordinates of the location, where the maneuver starts.

shape: {*nokia.maps.geo.Shape*}

This property holds an object representing the shape of the route segment following the given maneuver as a polyline.

time: {Number}

here

This property holds a value indicating the time when the given maneuver is to occur. For public transport maneuvers, it denotes the arrival or departure time according to the line's schedule. For private transport routes that have been calculated based on an explicit arrival or departure time, the value denotes the predicted time of the maneuver. The value reflects time specific to the time zone of the location of the maneuver.

toLink: {[nokia.maps.advrouting.RouteLink](#)}

This property holds a reference to an object representing the next link on the route.

travelTime: {Number}

This property holds the time needed to travel the length of the segment following the given maneuver until the next maneuver. Depending on the routing type, the value may reflect traffic conditions (traffic information).

Interface: Mode

This interface is a member of [nokia.maps.advrouting](#).

Interface Summary

This interface defines modes for route calculations.

[For full details, see [nokia.maps.advrouting.Mode](#)]

Table 28: Property Summary

Properties
features: {HashMap} This property is a hashmap of weighted route features to be applied when calculating the route.
trafficMode: {String} This property specifies if traffic information is to be considered when calculating the route.
transportModes: {String[]} This property holds an array of strings that identify the transport mode relevant to the given route calculation.
type: {String} This property holds the route type applicable to the given route calculation.

here

Interface Description

This interface defines modes for route calculations.

Property Details

features: {HashMap}

This property is a hashmap of weighted route features to be applied when calculating the route. The hash map associates route features (`RouteFeature`) with weighting indicators (`RouteFeatureWeight`). The possible values of `RouteFeature` are:

- "tollroad" - indicates toll roads
- "motorway" - indicates motorways
- "railFerry" - indicates rail ferries
- "boatFerry"- indicates boat ferries
- "publicTransport"- indicates public transport
- "tunnel"- indicates tunnels
- "dirtRoad"- indicates dirt roads
- "park"- indicates links through parks
- "HOVLane"- indicates high-occupancy vehicle (HOV) lanes
- "stairs"- indicates stairs (applicable to pedestrian routing only)

The possible values of `RouteFeatureWeight` are given in the list below. Please note that in versions of the API where the string values are not supported, the {Number} equivalent should be used:

- "strictExclude" ({Number} equivalent -3) - indicates that strictly excluded features are not part of the calculated route; if this condition cannot be fulfilled, no route is calculated; "HOVLanes" and "stairs" are examples for features, where a strict exclusion might be required
- "softExclude" ({Number} equivalent -2) - indicates that the corresponding feature is to be part of the calculated route, but if no route can be found because of these limitations, the condition is weakened
- "avoid" ({Number} equivalent -1) - indicates that the corresponding feature is to be avoided (the route is to avoid areas, road links, etc. with this feature)
- "normal" ({Number} equivalent 0) - indicates that the corresponding feature is neither preferred nor avoided in the route calculation
- "prefer" ({Number} equivalent 1) - indicates that the corresponding feature is to be preferred (included where possible) in the calculated route

here

For example the property features can be set (as part of a Mode object) as follows: { ..., features: { tollroad: -1, railFerry: 0}}

trafficMode: {String}

This property specifies if traffic information is to be considered when calculating the route. The possible values are:

- "enabled" - indicates that dynamic traffic conditions are to be taken into account (current traffic traffic patterns, short term closures, long term closures)
- "disabled" - indicates that dynamic traffic conditions are not to be taken into account (only apply time restrictions and seasonal closures)
- "default" - indicates that traffic-related constraints applicable for the selected route type, transport mode, departure time and entitlements are automatically taken into consideration

transportModes: {String[]}

This property holds an array of strings that identify the transport mode relevant to the given route calculation. The possible values are:

- "car" - indicates the route is to be suitable for cars
- "pedestrian" - indicates that the route is to be suitable for pedestrians
- "truck" - indicates that the route is to be suitable for trucks

type: {String}

This property holds the route type applicable to the given route calculation. The possible values are:

- "shortest" - indicates the shortest possible route in terms of distance traveled between start and destination, disregarding any traffic conditions
- "fastest" - indicates the shortest possible route in terms of travel time between the start and destination; the selected traffic mode determines whether the estimated travel time reflects traffic conditions
- "fastestNow" - indicates the fastest route at the current time, in other words, it combines the route type "fastest" with the traffic mode "enabled", and takes into consideration the departure time (if departure time is omitted, the current time, "now", is assumed)
- "directDrive" - indicates the fastest route at the time time of departure, without taking traffic conditions into consideration, in other words, it combines the route type "fastest" with

here

traffic mode "disabled", and takes into consideration the departure time (if departure time is omitted, the current time, "now" is assumed)

- "scenic" - indicates a route that favors scenic roads and landscapes

Interface: NavigationWaypointParameter

This interface is a member of [nokia.maps.advrouting](#).

Interface Summary

This interface represents a waypoint based on navigational information such as link positions.

[For full details, see [nokia.maps.advrouting.NavigationWaypointParameter](#)]

Table 29: Property Summary

Properties
<code>displayPosition</code> : { nokia.maps.geo.Coordinate } This property holds the geographic coordinates indicating where the waypoint is to be displayed on the map.
<code>linkPositions</code> : { nokia.maps.advrouting.LinkPosition []} This property holds a list of navigation positions of the waypoint represented by a position on a route link.
<code>streetPositions</code> : { nokia.maps.advrouting.StreetPosition []} This property holds a list of navigation positions of the waypoint represented by the geographic coordinates and a street name.

Interface Description

This interface represents a waypoint based on navigational information such as link positions. In a common use case, the user specifies a waypoint by selecting a place or a location after having executed a search. A valid navigation waypoint should at least contain property `linkPosition` or `streetPosition`.

Property Details

`displayPosition`: {[nokia.maps.geo.Coordinate](#)}

This property holds the geographic coordinates indicating where the waypoint is to be displayed on the map. It usually denotes the center of the location and is not navigable, that is it is not located on a link in the routing network in contrast to the navigation positions of a location.

`linkPositions`: {[nokia.maps.advrouting.LinkPosition](#)[]}

here

This property holds a list of navigation positions of the waypoint represented by a position on a route link. As there might be multiple ways to reach a waypoint (for example an airport with different entries and exits) the user can provide a list of navigation positions. Compared to street positions, link positions are the preferred way to identify navigation positions as they are more precise. Mixing mix street positions and link positions in one waypoint parameter is not supported.

streetPositions: {[nokia.maps.advrouting.StreetPosition](#)[][]}

This property holds a list of navigation positions of the waypoint represented by the geographic coordinates and a street name. As there might be multiple ways to reach a waypoint (for example an airport with different entries and exits) the user can provide a list of navigation positions. This property should only be used if no link information is available; [LinkPositions](#) are the preferred way to identify navigation positions, because they are more precise. Mixing street positions and link positions in one waypoint parameter is not supported.

Interface: Route

This interface is a member of [nokia.maps.advrouting](#).

Interface Summary

This interface defines a route.

[For full details, see [nokia.maps.advrouting.Route](#)]

Table 30: Property Summary

Properties
boundingBox: {nokia.maps.geo.BoundingBox} This property holds a bounding box object that enclosed the route.
legs: {nokia.maps.routing.RouteLeg[][]} This property holds an array of route legs (section of the route between any two consecutive waypoints).
mode: {nokia.maps.advrouting.Mode} This property holds the identifier of the mode used in the route calculation.
notes: {nokia.maps.advrouting.RouteNote} This property holds notes that are either related to the route calculation (routing options that could not be honored/applied) or that refer the route as a whole.
routeId: {String} This property holds a unique identifier of the route.

here

Properties

shape: {*nokia.maps.geo.Shape*}

This property holds the shape of the route as a polyline.

summary: {*nokia.maps.advrouting.RouteSummary*}

This property holds a reference to a RouteSummary object.

summaryByCountry: {*nokia.maps.advrouting.RouteSummaryByCountry*[]}

This optional property is designed to hold an array of route summaries by country.

waypoints: {*nokia.maps.advrouting.Waypoint*[]}

This property holds a list of waypoints that were defined as part of the route calculation request.

Table 31: Method Summary

Methods

getDestination () : {*nokia.maps.advrouting.Waypoint*}

This method retrieves the last waypoint of the route (the end of the route).

getManeuvers () : {*nokia.maps.advrouting.Maneuver*[]}

This method retrieves an array containing all the maneuvers included in the route.

getStart () : {*nokia.maps.advrouting.Waypoint*}

This method retrieves the first waypoint of the given route (the start of the route).

Interface Description

This interface defines a route. A route describes a path along which a person can walk or a vehicle can travel. It has a starting point and a destination and it can pass through any number of intermediate (or "via") points. The start and destination points of a route as well as the via points are referred to as waypoints. A section of a route between any two consecutive waypoints is a leg of the route (or "route leg").

Property Details

boundingBox: {*nokia.maps.geo.BoundingBox*}

This property holds a bounding box object that enclosed the route.

legs: {*nokia.maps.routing.RouteLeg*[]}

This property holds an array of route legs (section of the route between any two consecutive waypoints).

here

mode: {*nokia.maps.advrouting.Mode*}

This property holds the identifier of the mode used in the route calculation.

notes: {*nokia.maps.advrouting.RouteNote*}

This property holds notes that are either related to the route calculation (routing options that could not be honored/applied) or that refer the route as a whole. In addition to these notes, further notes can be attached to maneuvers. The maneuver notes are typically related to the route segment following the maneuver.

routeId: {String}

This property holds a unique identifier of the route. It is calculated from the route links and can be used to reproduce any previously calculated route. Thus, the value of this property is not a temporary ID, but can be used as persistent reference to the route.

shape: {*nokia.maps.geo.Shape*}

This property holds the shape of the route as a polyline. The accuracy depends on the resolution specified in mpp (meters per pixel) when requesting the route. In some use cases (for example Web portals), only the shape of the route is required, without the complete nested structure of the entire route and details of the links. In such cases, the shape is not acquired by traversing the route links, but can be represented using this attribute at route level.

summary: {*nokia.maps.advrouting.RouteSummary*}

This property holds a reference to a RouteSummary object. The use of this property is optional.

summaryByCountry: {*nokia.maps.advrouting.RouteSummaryByCountry*[][]}

This optional property is designed to hold an array of route summaries by country.

waypoints: {*nokia.maps.advrouting.Waypoint*[][]}

here

This property holds a list of waypoints that were defined as part of the route calculation request. The first waypoint defines the start of the route, the last marks the destination. Any points in between the two are via points.

Method Details

getDestination(): `{nokia.maps.advrouting.Waypoint}`

This method retrieves the last waypoint of the route (the end of the route).

Returns:

`{nokia.maps.advrouting.Waypoint}`

An object representing the end point of the route.

getManeuvers(): `{nokia.maps.advrouting.Maneuver[]}`

This method retrieves an array containing all the maneuvers included in the route.

Returns:

`{nokia.maps.advrouting.Maneuver[]}`

An array of objects representing the maneuvers that are part of the given route

getStart(): `{nokia.maps.advrouting.Waypoint}`

This method retrieves the first waypoint of the given route (the start of the route).

Returns:

`{nokia.maps.advrouting.Waypoint}`

An object representing the start of the route.

Interface: RouteLink

This interface is a member of `nokia.maps.advrouting`.

Interface Summary

This interface represents a route link.

here

[For full details, see [nokia.maps.advrouting.RouteLink](#)]

Table 32: Property Summary

Properties
additionalData: {Object} This property is a generic container to store additional information about the link.
length: {Number} This property holds the length of the link.
linkId: {String} This property holds a unique identifier of a route link.
nextLink: {nokia.maps.advrouting.RouteLink} This property holds a reference to the next link on the route.
remainDistance: {Number} This property holds a value reflecting the distance from the start of the given link to the destination point of the route.
remainTime: {Number} This property holds the predicted travel time from the start of the given link to the destination of the route.
shape: {nokia.maps.geo.Shape} This property holds a polyline that represents the shape of the link.

Interface Description

This interface represents a route link. A route link is the smallest part of a navigable route that is uninterrupted by road junctions, crossroads, forks, roundabouts (rotaries), etc. Note that one or more route links make up a route segment ([nokia.maps.advrouting.RouteSegment](#)).

Property Details

`additionalData`: {Object}

This property is a generic container to store additional information about the link.

`length`: {Number}

This property holds the length of the link.

`linkId`: {String}

here

This property holds a unique identifier of a route link. A minus ("-") at the beginning of the identifier indicates that the link is traversed from node to reference node. An unsigned identifier means that the link is traversed from reference node to node.

nextLink: {[*nokia.maps.advrouting.RouteLink*](#)}

This property holds a reference to the next link on the route.

remainDistance: {Number}

This property holds a value reflecting the distance from the start of the given link to the destination point of the route.

remainTime: {Number}

This property holds the predicted travel time from the start of the given link to the destination of the route. Note that traffic information is taken into account if the user is authorized to use traffic information.

shape: {[*nokia.maps.geo.Shape*](#)}

This property holds a polyline that represents the shape of the link.

Interface: RouteNote

This interface is a member of [*nokia.maps.advrouting*](#).

Interface Summary

This interface represents route notes, which provide additional information about a route.

[For full details, see [*nokia.maps.advrouting.RouteNote*](#)]

Table 33: Property Summary

Properties
<i>additionalData</i>: {Object} This property holds a container object for additional data to be stored along with the note. <i>linkIds</i>: {String[]} This property holds an array of ids of route links to which the note applies.

here

Properties

position: {*nokia.maps.geo.Coordinate*}

This property holds the geographic coordinates of the position along the route to which the note applies.

text: {String}

This property holds a short piece of text describing the note.

validityPeriod: {String}

This property holds a string value indicating the validity period of the note.

Interface Description

This interface represents route notes, which provide additional information about a route. The notes can either relate to the calculation itself (such as unused routing options) or to the characteristics of the route (for example, entering a toll road, passing a border, etc.)

Property Details

additionalData: {Object}

This property holds a container object for additional data to be stored along with the note.

linkIds: {String[]}

This property holds an array of ids of route links to which the note applies.

position: {*nokia.maps.geo.Coordinate*}

This property holds the geographic coordinates of the position along the route to which the note applies.

text: {String}

This property holds a short piece of text describing the note. Please note that the property is not subject to internationalization and should therefore not be used in user displays.

validityPeriod: {String}

This property holds a string value indicating the validity period of the note. This attribute is used to specify time-dependent restrictions (for example "road is closed during winter").

here

Interface: RouteRepresentationOptions

This interface is a member of [nokia.maps.advrouting](#).

Interface Summary

This interface defines options that can be used to fine-tune the scope and granularity of a route representation in the response.

[For full details, see [nokia.maps.advrouting.RouteRepresentationOptions](#)]

Table 34: Property Summary

Properties
instructionFormat: {String} This property holds the text formatting specification for instructions associated with a maneuver.
language: {String} This property holds the identifier of the language to be used for all textual information.
legAttributes: {String[]} This property holds a list of strings in the format "[-]attribute", where "attribute" denotes an identifier of a route leg attribute.
linkAttributes: {String[]} This property holds a list of strings that identify which attributes are to be included in route links.
maneuverAttributes: {String[]} This property holds a list of strings that identify which attributes are to be included in route maneuvers.
representationMode: {String} This property holds a value indicating the representation mode for the route.
routeAttributes: {String[]} This property holds a list of route attribute names in the format "[-]attribute" that are included in or excluded from the returned routes.
viewBounds: {nokia.maps.geo.BoundingBox} This property holds an object defining the route view bounds.
viewResolution: {Number} This property holds a value indicating the resolution of the route view in meters per pixel (mpp), allowing for the route shape in the response to reflect the resolution of the client.

here

Interface Description

This interface defines options that can be used to fine-tune the scope and granularity of a route representation in the response.

Property Details

instructionFormat: {String}

This property holds the text formatting specification for instructions associated with a maneuver.

- "text" - returns the instruction as a plain text string
- "native" - based on the message templates used in the routing engine with the different parameter values being tagged with corresponding xml tags.
- "html" - instruction format is based on span tags with different CSS classes to assign semantics to the tagged part of the instruction

language: {String}

This property holds the identifier of the language to be used for all textual information. The value is a string holding a two-letter language identifier (ISO 639-1), followed by a dash, followed by a two-letter country code (where appropriate), for example "en-us". The supported languages are:

- en-uk
- en-us
- fr-fr
- de-de
- es-es
- it-it

legAttributes: {String[]}

This property holds a list of strings in the format "[-]attribute", where "attribute" denotes an identifier of a route leg attribute. If the list is combined with the value "all", "none" or a representation mode value (see [nokia.maps.advrouting.RouteRepresentationOptions#representationMode](#)), the listed attributes are interpreted as delta requests and a "-" before an attribute identifier indicates the attribute is to be removed, while no prefix indicates that the attribute is to be added.

If the property is not specified, the assumed defaults are "maneuvers", "waypoint" and "length".

here

The list can include the following strings:

- "waypoint" - indicates whether waypoints are to be included in the route leg
- "maneuvers" - indicates whether the maneuvers of the route leg are to be included
- "links" - indicates whether the links along the route leg are to be included; if "links" is not specified, only route segments are included
- "length" - indicates whether the route leg should include its length
- "travelTime" - indicates whether the route leg should include its duration

linkAttributes: {String[]}

This property holds a list of strings that identify which attributes are to be included in route links. If the property is not specified, the default list includes "shape", "speedLimit", "dynamicSpeedInfo", "address". The list can include the following strings:

- "shape" - indicates whether the link should include its geometry
- "length" - indicates whether the link should include its length
- "speedLimit" - indicates whether the link should include the applicable speed limit
- "dynamicSpeedInfo" - indicates whether the link should include dynamic speed information
- "incidents" - indicates whether the link should include incidents
- "truckRestrictions" - indicates whether the link should include truck restrictions
- "externalResources" - indicates whether the link should include external resources
- "flags" - indicates whether the link should include link flags
- "corridorLevel" - indicates whether the link should include the corridor level
- "nextLink" - indicates whether the link should include the next link along the route
- "stubs" - indicates whether the link should include the corridor stubs
- "publicTransportLine" - indicates whether the link should include information about public transport line(s)
- "TMCCodes" - indicates whether the link should include information about the covered TMC Codes
- "jamFactor" - indicates whether the link's dynamic speed info should include information about the jam factor
- "jamFactorTrend" - indicates whether the link's dynamic speed info should include information about the jam factor trend
- "confidence" - indicates whether the link's dynamic speed info should include information about the level of confidence regarding traffic information

here

- "address" - indicates whether the link should include the link's address
- "roadNumber" - indicates whether the link should include the link's road number
- "roadName" - indicates whether the link should include the link's road number
- "freewayJunction" - indicates whether the link should include the name of the freeway junction
- "timezone" - indicates whether the link should include the timezone
- "remainTime" - indicates whether the link should include information about the remaining time until the destination is reached.
- "remainDistance" - indicates whether the link should include information about the remaining distance until the destination is reached.
- "maneuver" - indicates whether the link should include information about the associated maneuver.
- "functionalClass" - indicates whether the link should include information about the functional class.
- "nextStopName" - indicates whether the link should include information about the next stop.
- "additionalData" - indicates whether the link should include the container for additional data.
- "speedCategory" - indicates whether the link should include speed category information.
- "predecessors" - indicates whether the link should include predecessor link ID information.
- "successors" - indicates whether the link should include predecessor link ID information.
- "indices" - indicates whether shape index information (FirstPoint, LastPoint) should be included in links instead of copying shape points.

maneuverAttributes: {String[]}

This property holds a list of strings that identify which attributes are to be included in route maneuvers. If the property is not specified, the default list includes "position" and "link". The list can include the following strings:

- "position" - indicates whether the position should be included in each maneuvers
- "shape" - indicates whether the shape of the segment to the next maneuver should be included in the maneuvers
- "travelTime" - indicates whether the time needed to the next maneuver should be included in the maneuvers
- "length" - indicates whether the distance to the next maneuver should be included in the maneuvers

here

- "time" - indicates whether the point in time when the maneuver will take place should be included in the maneuvers
- "link" - indicates whether the link, where the maneuver takes place is to be included
- "publicTransportLine" - indicates whether the information about the public transport line(s) should be included
- "platform" - indicates whether the platform information about platform related to a public transport line should be included
- "equipment" - indicates whether equipment at the maneuver should be included in the maneuvers
- "lane" - indicates whether lane information should be included
- "roadName" - indicates whether the road name should be included in the maneuvers
- "nextRoadName" - indicates whether the name of the next road shall be included in the maneuvers
- "roadNumber" - indicates whether the road number should be included in the maneuvers
- "nextRoadNumber" - indicates whether the number of the next road should be included in the maneuvers
- "roadTemplate" - indicates whether the template for route display should be included in the maneuvers
- "signPost" - indicates whether sign post information should be included
- "notes" - indicates whether additional notes should be included in the maneuvers
- "action" - indicates whether actions should be included in the maneuvers
- "direction" - indicates whether directions should be included in the maneuvers
- "nextManeuver" - indicates whether a reference to the next maneuver should be included in the maneuvers
- "freewayExit" - indicates whether the freeway exit should be included in the maneuvers
- "freewayJunction" - indicates whether the freeway junction should be included in the maneuvers
- "baseTime" - indicates whether the BaseTime information should be included in the maneuvers. By default, BaseTime information is not included in the maneuvers
- "trafficTime" - indicates whether the TrafficTime information should be included in the maneuvers. By default, TrafficTime information is not included in the maneuvers
- "indices" - indicates whether shape index information (FirstPoint, LastPoint) should be included in the maneuvers instead of copying shape points to the maneuver
- "waitTime" - indicates whether wait time information should be included in public transport maneuvers

representationMode: {String}

This property holds a value indicating the representation mode for the route. It applies to the use cases, where not all objects and attributes in the route object model are required at once, and allows the API user to specify which part of the route is to be returned. The possible values are:

- "overview" - indicates the overview mode only, the returned data include the [nokia.maps.advrouting.Route](#) and [nokia.maps.advrouting.RouteSummary](#) objects
- "dragNDrop" - indicates the drag-and-drop mode to be used during drag and drop (re-routing) operations. The response contain only the shape of the route restricted to the view bounds provided in the representation options
- "navigation" - indicates the navigation mode to provide all information necessary to support a navigation device. This mode activates the most extensive data response that includes all links to allow detailed display, while navigating
- "turnByTurn" - indicates the turn-by-turn mode to provide all the information necessary to support turn-by-turn navigation. This mode activates all data needed for navigation, excluding any redundancies

REST routing API also supports following options:

- "display" - allows to show the route with all maneuvers. Links won't be included in the response
- "linkPaging" - is used when incrementally loading links while navigating along the route. The response will be limited to link information.

routeAttributes: {String[]}

This property holds a list of route attribute names in the format "[~]attribute" that are included in or excluded from the returned routes. If the property is not specified, the default attributes are "waypoints", "summary", "maneuvers" and "links".

Note that a list of attributes in combination with "all", "none" or items from [nokia.maps.advrouting.RouteRepresentationOptions#representationMode](#) is treated as a delta specification, where a minus sign ("-") preceding an attribute name indicates the attribute is to be removed/excluded, while an attribute without a minus sign is included/added.

The list of attributes can include the following strings:

- "segments" - indicates whether are to be included in the route
- "waypoints" - indicates whether via points are to be included in the route
- "summary" - indicates whether a route summary is to be provided for the route

here

- "summaryByCountry" - indicates whether a country-based route summary is to be provided for the route
- "shape" - indicates whether the shape of the route is to be provided for the route
- "boundingBox" - indicates whether the bounding box of the route is to be provided for the route
- "notes" - indicates whether route route notes are to be included in the route
- "none" - indicates that no attributes are to be provided for the route; "none" can be used in combination with a list of attributes to be added, first to clear all attributes (including the default ones) and then to add only those required, thus creating a customized list of attributes to be acted upon
- "all" - indicates that all attributes are to be provided for the route; "all" can be combined with a list of attribute names preceded by the minus sign ("-") to indicate the attributes to be excluded, which in effect means "add all, then remove those preceded by a minus sign (for example, ["all", "-summary"])
- members of [nokia.maps.advrouting.RouteRepresentationOptions#representationMode](#), followed by a list of zero or more attributes to include/exclude

viewBounds: {[nokia.maps.geo.BoundingBox](#)}

This property holds an object defining the route view bounds. If view bounds are specified in the request, only shapes and links which fit into these bounds are returned. A typical use case for the use of this property is a drag-and-drop scenario, where the client requires only a rough visual update of the route within the currently visible bounds.

viewResolution: {Number}

This property holds a value indicating the resolution of the route view in meters per pixel (mpp), allowing for the route shape in the response to reflect the resolution of the client.

Interface: RouteResponseMetaInfo

This interface is a member of [nokia.maps.advrouting](#).

Interface Summary

This interface represents meta information which is not directly related to the route calculation.

[For full details, see [nokia.maps.advrouting.RouteResponseMetaInfo](#)]

here

Table 35: Property Summary

Properties
<code>additionalData</code>: {Object} This property holds a container for additional data to be stored along with the response meta info.
<code>requestId</code>: {String} This property holds the request id mirrored from the route calculation request structure and can be used to trace back requests.
<code>timestamp</code>: {Number} This property holds the time at which the route calculation was performed.

Interface Description

This interface represents meta information which is not directly related to the route calculation.

Property Details

`additionalData`: {Object}

This property holds a container for additional data to be stored along with the response meta info.

`requestId`: {String}

This property holds the request id mirrored from the route calculation request structure and can be used to trace back requests.

`timestamp`: {Number}

This property holds the time at which the route calculation was performed.

Interface: RouteSegment

This interface is a member of [*nokia.maps.advrouting*](#).

Interface Summary

This interface represents a portion of a route between two maneuvers.

[For full details, see [*nokia.maps.advrouting.RouteSegment*](#)]

here

Table 36: Property Summary

Properties
<code>additionalData</code>: {Object} This property is a generic container that stores additional information regarding the given route segment.
<code>alternativeSegments</code>: {<code>nokia.maps.advrouting.RouteSegment</code>[]} This property holds an array of objects representing alternative route segments.
<code>length</code>: {Number} This property holds the length of the route link.
<code>maneuver</code>: {<code>nokia.maps.advrouting.Maneuver</code>} This property holds an object representing the maneuver related to the given route segment.
<code>nextSegment</code>: {<code>nokia.maps.advrouting.RouteSegment</code>} This property holds a reference to the next segment along the route.
<code>shape</code>: {<code>nokia.maps.geo.Shape</code>} This property holds a polyline that defines the shape of the route segment.
<code>travelTime</code>: {Number} This property holds a value representing the length of time required to travel the length of the given route segment.

Interface Description

This interface represents a portion of a route between two maneuvers. A route segment is the smallest sub-division of a calculated route, unless the route attributes ([`nokia.maps.advrouting.RouteRepresentationOptions#routeAttributes`](#)) include "links", in which case each route segment includes one or more instances of [`nokia.maps.advrouting.RouteLink`](#). Maneuvers may be associated with a route segment and contain instructions on how to proceed from one route segment to another.

Property Details

`additionalData`: {Object}

This property is a generic container that stores additional information regarding the given route segment.

`alternativeSegments`: {[`nokia.maps.advrouting.RouteSegment`](#)[]}

This property holds an array of objects representing alternative route segments.

here

length: {Number}

This property holds the length of the route link.

maneuver: {[nokia.maps.advrouting.Maneuver](#)}

This property holds an object representing the maneuver related to the given route segment.

nextSegment: {[nokia.maps.advrouting.RouteSegment](#)}

This property holds a reference to the next segment along the route.

shape: {[nokia.maps.geo.Shape](#)}

This property holds a polyline that defines the shape of the route segment.

travelTime: {Number}

This property holds a value representing the length of time required to travel the length of the given route segment. Depending on the routing type, traffic information is taken into consideration when calculating this value.

Interface: RouteSummary

This interface is a member of [nokia.maps.advrouting](#).

Interface Summary

This interface represents a route summary.

[For full details, see [nokia.maps.advrouting.RouteSummary](#)]

Table 37: Property Summary

Properties
baseTime: {Number} This property holds the total route travel time, taking into account the mode of transport, but not traffic information.
distance: {Number} This property holds the total travel distance (the length of the route).
entries: {nokia.maps.advrouting.RouteSummaryEntry}

here

Properties

This property holds an array of route summary entries.

flags: {String[]}

This property holds an array of text strings that identify special link characteristics (such as ferry or motorway usage) covered by the route.

trafficTime: {Number}

This property holds the total route travel time, taking into account traffic information.

Interface Description

This interface represents a route summary.

Property Details

baseTime: {Number}

This property holds the total route travel time, taking into account the mode of transport, but not traffic information.

distance: {Number}

This property holds the total travel distance (the length of the route).

entries: {*nokia.maps.advrouting.RouteSummaryEntry*}

This property holds an array of route summary entries.

flags: {String[]}

This property holds an array of text strings that identify special link characteristics (such as ferry or motorway usage) covered by the route. The possible values are:

- "motorway" - indicates that the link is a motorway
- "boatFerry" - indicates that the link can only be traversed by using a boat ferry
- "railFerry" - indicates that the link can only be traversed by using a rail ferry
- "publicTransport" - indicates that the link can only be traversed by using public transport
- "gatedArea" - indicates that the link is part of a gated area
- "privateRoad" - indicates that the link is part of a private road
- "tollroad" - indicates that the link is part of a toll road

here

- "station" - indicates a station

trafficTime: {Number}

This property holds the total route travel time, taking into account traffic information.

Interface: RouteSummaryByCountry

This interface is a member of [nokia.maps.advrouting](#).

Interface Summary

This interface represents a summary of a route for a specific country.

[For full details, see [nokia.maps.advrouting.RouteSummaryByCountry](#)]

Table 38: Property Summary

Properties
country: {String} This property holds the country code. The value is a string holding a two-letter language identifier (ISO 639-1), followed by a dash, followed by a two-letter country code (where appropriate), for example "en-us".

Directly Inherited Properties

Inherited from class [nokia.maps.advrouting.RouteSummary](#) :

[baseTime](#), [distance](#), [entries](#), [flags](#), [trafficTime](#)

Interface Description

This interface represents a summary of a route for a specific country.

Property Details

country: {String}

This property holds the country code. The value is a string holding a two-letter language identifier (ISO 639-1), followed by a dash, followed by a two-letter country code (where appropriate), for example "en-us".

Interface: RouteSummaryEntry

This interface is a member of [nokia.maps.advrouting](#).

here

Interface Summary

This interface defines an entry in a route summary.

[For full details, see [nokia.maps.advrouting.RouteSummaryEntry](#)]

Table 39: Property Summary

Properties
distance: {Number} This property holds the travel distance covered by the part of the route to which the summary entry relates. label: {String} This property holds a textual description of the summary entry.

Interface Description

This interface defines an entry in a route summary. A route summary contains a description of the route limited to the essential streets and maneuvers. Each of these maneuvers/routes is described in a route summary entry.

Property Details

[distance](#): {Number}

This property holds the travel distance covered by the part of the route to which the summary entry relates.

[label](#): {String}

This property holds a textual description of the summary entry.

Interface: StreetPosition

This interface is a member of [nokia.maps.advrouting](#).

Interface Summary

This interface defines a navigation position on the basis of geographic coordinates and a street name.

[For full details, see [nokia.maps.advrouting.StreetPosition](#)]

here

Table 40: Property Summary

Properties
position: {<i>nokia.maps.geo.Coordinate</i>} This property holds an object containing the geographic coordinates of the location. streetName: {String} This property holds the name of the street.

Interface Description

This interface defines a navigation position on the basis of geographic coordinates and a street name. The property containing the geographic coordinates is mandatory, while the property that holds the street name is optional, but when set, helps select the correct street/road in complex intersection scenarios (for example, a bridge crossing another road).

Property Details

position: {[*nokia.maps.geo.Coordinate*](#)}

This property holds an object containing the geographic coordinates of the location.

streetName: {String}

This property holds the name of the street. This name allows to distinguish two streets crossing a the location given by `position`, potentially separated by a bridge/tunnel. The property can contain the street/road name or the road number (if no name is available).

Interface: TravelProgress

This interface is a member of [*nokia.maps.advrouting*](#).

Interface Summary

This interface encapsulates information reflecting progress along the route in a navigation scenario.

[For full details, see [*nokia.maps.advrouting.TravelProgress*](#)]

Table 41: Property Summary

Properties
mappedPosition: {<i>nokia.maps.routing.Waypoint</i>}

here

Properties

This property holds the current position, potentially evaluated after map matching a sequence of positions provided in the request.

remainDistance: {Number}

This property holds the remaining distance from the current position to the destination.

remainTime: {Number}

This property holds the remaining journey time, the time needed to travel from the current position to the destination.

Interface Description

This interface encapsulates information reflecting progress along the route in a navigation scenario.

Property Details

mappedPosition: {*nokia.maps.routing.Waypoint*}

This property holds the current position, potentially evaluated after map matching a sequence of positions provided in the request.

remainDistance: {Number}

This property holds the remaining distance from the current position to the destination.

remainTime: {Number}

This property holds the remaining journey time, the time needed to travel from the current position to the destination. The value takes into consideration traffic information.

Interface: TruckProfile

This interface is a member of *nokia.maps.advrouting*.

Interface Summary

This interface represents a truck profile for use in route calculations.

[For full details, see *nokia.maps.advrouting.TruckProfile*]

here

Table 42: Property Summary

Properties
hasTrailer: {Boolean} This property holds a Boolean value indicating whether the truck has a trailer (true) or not (false).
height: {Number} This property holds a value that indicates the height of the truck in meters.
length: {Number} This property holds a value that indicates the length of the truck in meters.
limitedWeight: {Number} This property holds a value that indicates the maximum allowed weight of the vehicle in tons.
permittedGrossWeight: {Number} This property holds a value that indicates the permitted gross weight of the vehicle.
shippedHazardousGoods: {string[]} This property holds a list of hazardous goods that carried by the truck.
trailerWeight: {Number} This property holds a value that indicates the weight of the trailer attached to the vehicle.
weightPerAxle: {Number} This property holds a value that indicates the weight of the vehicle per axle.
width: {Number} This property holds a value that indicates the width of the truck in meters.

Interface Description

This interface represents a truck profile for use in route calculations.

Property Details

hasTrailer: {Boolean}

This property holds a Boolean value indicating whether the truck has a trailer (true) or not (false).

height: {Number}

This property holds a value that indicates the height of the truck in meters.

length: {Number}

here

This property holds a value that indicates the length of the truck in meters.

limitedWeight: {Number}

This property holds a value that indicates the maximum allowed weight of the vehicle in tons.

permittedGrossWeight: {Number}

This property holds a value that indicates the permitted gross weight of the vehicle.

shippedHazardousGoods: {string[]}

This property holds a list of hazardous goods that carried by the truck. The possible values are:

- "all" - indicates all recognized types of hazardous goods
- "explosive" - indicates explosive materials
- "gas" - indicates gas
- "flammable" - indicates flammable materials
- "combustible" - indicates combustible materials
- "organic" - indicates organic materials
- "poison" - indicates poisonous materials
- "radioActive" - indicates radioactive materials
- "corrosive" - indicates corrosive materials
- "poisonousInhalation" - indicates goods which are poisonous upon inhalation
- "harmfulToWater" - indicates goods which are harmful to water
- "other" - indicates other types of hazardous goods

trailerWeight: {Number}

This property holds a value that indicates the weight of the trailer attached to the vehicle.

weightPerAxle: {Number}

This property holds a value that indicates the weight of the vehicle per axle.

width: {Number}

here

This property holds a value that indicates the width of the truck in meters.

Interface: Waypoint

This interface is a member of [nokia.maps.advrouting](#).

Interface Summary

This interface defines a waypoint, which is a point along a route explicitly specified as part of route definition as a location through which the route must pass.

[For full details, see [nokia.maps.advrouting.Waypoint](#)]

Table 43: Property Summary

Properties
linkId : {String} This property holds the id of the link to which the waypoint belongs.
mappedPosition : { nokia.maps.geo.Coordinate } This property holds an object containing the geographical coordinates of the nearest point on the link (route) to the original position specified as part of the route calculation request (the position of the waypoint on the calculated route).
matchQuality : {Number} Defines the quality of the map matching results, between 0.0 and 1.0, where 1.0 represents a 100% match.
originalPosition : { nokia.maps.geo.Coordinate } This property holds an object containing the geographical coordinates of the position of the waypoint as it was originally specified in the route calculation request.
spot : {Number} Contains the relative position of the mapped location along the link, as the fractional distance between the link's reference node and the non-reference node.
type : {String} Defines the type of the waypoint, either stopOver or passThrough.

Interface Description

This interface defines a waypoint, which is a point along a route explicitly specified as part of route definition as a location through which the route must pass. A waypoint can correspond to a route link position ([nokia.maps.advrouting.LinkPosition](#)) or it can be the result of map matching. In the first case, the attribute mappedPosition is not filled.

here

Property Details

linkId: {String}

This property holds the id of the link to which the waypoint belongs.

mappedPosition: {[nokia.maps.geo.Coordinate](#)}

This property holds an object containing the geographical coordinates of the nearest point on the link (route) to the original position specified as part of the route calculation request (the position of the waypoint on the calculated route).

matchQuality: {Number}

Defines the quality of the map matching results, between 0.0 and 1.0, where 1.0 represents a 100% match.

originalPosition: {[nokia.maps.geo.Coordinate](#)}

This property holds an object containing the geographical coordinates of the position of the waypoint as it was originally specified in the route calculation request.

spot: {Number}

Contains the relative position of the mapped location along the link, as the fractional distance between the link's reference node and the non-reference node. Ranges in value from 0 to 1.

type: {String}

Defines the type of the waypoint, either stopOver or passThrough.

Interface: WaypointParameter

This interface is a member of [nokia.maps.advrouting](#).

Interface Summary

This interface represents a route waypoint parameter.

[For full details, see [nokia.maps.advrouting.WaypointParameter](#)]

here

Interface Description

This interface represents a route waypoint parameter. Waypoints of a route can be defined by specifying a rough position ([nokia.maps.advrouting.GeoWaypointParameter](#)), an exact reference to a link ([nokia.maps.advrouting.NavigationWaypointParameter](#) with [nokia.maps.advrouting.LinkPositions](#)), a reference to a street ([nokia.maps.advrouting.NavigationWaypointParameter](#) with [nokia.maps.advrouting.StreetPositions](#)). The routing service tries to find the best matching link to be used based on the routing network. Each waypoint may be specified in either mode.

Namespace: advsearch

This namespace is a member of [nokia.maps](#).

Namespace Summary

This namespace defines classes that are required for extended search use cases.

Namespace Description

This namespace defines classes that are required for extended search use cases.

Interface: Address

This interface is a member of [nokia.maps.advsearch](#).

Interface Summary

This interface defines an address of a location.

[For full details, see [nokia.maps.advsearch.Address](#)]

Table 44: Property Summary

Properties
additionalData : {Object} This property holds the generic key/value container to keep additional address attributes.
alternativeAttributes : { nokia.maps.advsearch.AlternativeValue } This property holds the attributes of the address in different languages and/or with different semantics.
city : {String} This property refers to the locality of the address.
country : {String}

here

Properties

This property holds an ISO 3166-alpha-3 country code.

county: {String}

This property holds the second address element below the country.

district: {String}

This property holds the first address element below the city.

floor: {String}

This property holds the floor number.

houseNumber: {String}

This property holds the house number.

label: {String}

This property holds the assembled address value for display purposes.

postalCode: {String}

This property holds the postal code.

state: {String}

This property holds the first part of the address below the country.

street: {String}

This property holds the street name in the requested primary language.

suite: {String}

This property holds the suite number or name.

Interface Description

This interface defines an address of a location. The attributes are normalized to US feature names and can be mapped to the local feature levels (for example, State matches "Bundesland" in Germany) using mapping tables for GDF.

Property Details

additionalData: {Object}

This property holds the generic key/value container to keep additional address attributes.

alternativeAttributes: {*nokia.maps.advsearch.AlternativeValue*}

This property holds the attributes of the address in different languages and/or with different semantics.

here

city: {String}

This property refers to the locality of the address. Some products map "city" directly to `Address.city`. For GDF, a country-specific mapping is required.

country: {String}

This property holds an ISO 3166-alpha-3 country code.

county: {String}

This property holds the second address element below the country. The use of this field is optional if the corresponding information is not available.

district: {String}

This property holds the first address element below the city. The use of this field is optional if the corresponding information is not available.

floor: {String}

This property holds the floor number.

houseNumber: {String}

This property holds the house number. Depending on regional requirements, it can hold house name. For API implementations which do not support separate street and house number fields, this field can be omitted.

label: {String}

This property holds the assembled address value for display purposes.

postalCode: {String}

This property holds the postal code.

here

state: {String}

This property holds the first part of the address below the country. Note that some product maps "region" directly to `Address.state`. For GDF, a country-specific mapping is required.

street: {String}

This property holds the street name in the requested primary language. For API implementations which do not support separate street and house number fields, this field may also contain the house number value.

suite: {String}

This property holds the suite number or name.

Interface: AlternativeValue

This interface is a member of [nokia.maps.advsearch](#).

Interface Summary

This interface defines a container for an alternative value of an attribute on a class such as [nokia.maps.advsearch.Address](#).

[For full details, see [nokia.maps.advsearch.AlternativeValue](#)]

Table 45: Property Summary

Properties
key: {String} This property holds the name of the attribute.
language: {String} This property identifies the language of the alternative text (via an RFC 5646 Language Tag).
semantics: {String} This property identifies the semantics of the alternative value.
type: {String} This property holds an identifier of the type of the alternative value.
value: {String} This property holds the text value in the selected language and with the selected semantics

here

Interface Description

This interface defines a container for an alternative value of an attribute on a class such as [nokia.maps.advsearch.Address](#). The container is generic and designed to hold textual values for any attribute in different languages and/or with different semantics.

Property Details

key: {String}

This property holds the name of the attribute.

language: {String}

This property identifies the language of the alternative text (via an RFC 5646 Language Tag).

semantics: {String}

This property identifies the semantics of the alternative value. The possible values are:

- "synonym"
- "exonym"
- "unclassified"

type: {String}

This property holds an identifier of the type of the alternative value. The possible values are:

- "baseName"
- "shortBaseName"
- "abbreviation"

value: {String}

This property holds the text value in the selected language and with the selected semantics

Interface: Category

This interface is a member of [nokia.maps.advsearch](#).

here

Interface Summary

This interface defines a Point of Interest (POI) Category.

[For full details, see [nokia.maps.advsearch.Category](#)]

Table 46: Property Summary

Properties
<code>alternativeNames</code>: {nokia.maps.advsearch.AlternativeValue[]} This property holds an array of alternative names of the given category in different languages and/or with different semantics.
<code>categoryId</code>: {String} This property uniquely identifies the category within the named category system.
<code>categorySystemId</code>: {String} This property uniquely identifies the category system to which the category belongs.
<code>description</code>: {String} This property holds a description for the given category.
<code>name</code>: {String} This property holds a descriptive name of the category for display purposes.

Interface Description

This interface defines a Point of Interest (POI) Category. Different categories can be assigned to places. To support different types of categorization, a category always refers to a certain category system.

Property Details

`alternativeNames`: {[nokia.maps.advsearch.AlternativeValue](#)[]}
 This property holds an array of alternative names of the given category in different languages and/or with different semantics.

`categoryId`: {String}
 This property uniquely identifies the category within the named category system.

`categorySystemId`: {String}

here

This property uniquely identifies the category system to which the category belongs.

description: {String}

This property holds a description for the given category.

name: {String}

This property holds a descriptive name of the category for display purposes.

Interface: CategoryFilter

This interface is a member of [nokia.maps.advsearch](#).

Interface Summary

This interface represents a search filter type that matches search results to certain categories.

[For full details, see [nokia.maps.advsearch.CategoryFilter](#)]

Table 47: Property Summary

Properties
categoryId: {String} This property uniquely identifies the category within the named category system.
categorySystemId: {String} This property uniquely identifies the category system to which the category belongs.
name: {String} This property holds the name of the given category filter.

Interface Description

This interface represents a search filter type that matches search results to certain categories.

Property Details

categoryId: {String}

This property uniquely identifies the category within the named category system.

here

categorySystemId: {String}

This property uniquely identifies the category system to which the category belongs.

name: {String}

This property holds the name of the given category filter.

Interface: Contact

This interface is a member of [nokia.maps.advsearch](#).

Interface Summary

This interface encapsulates contact information associated with a Place.

[For full details, see [nokia.maps.advsearch.Contact](#)]

Table 48: Property Summary

Properties
description: {String} This property holds a description of the communication channel (for example: "company's website", "phone number to be used in emergency").
type: {String} This property holds a communication channel type identifier.
value: {String} This property holds the phone/fax number, e-mail/IM address or URL as appropriate to the contact type.

Interface Description

This interface encapsulates contact information associated with a Place.

Property Details

description: {String}

This property holds a description of the communication channel (for example: "company's website", "phone number to be used in emergency").

type: {String}

here

This property holds a communication channel type identifier. The possible values are:

- "phone"
- "email"
- "URL"
- "fax"
- "IM"

value: {String}

This property holds the phone/fax number, e-mail/IM address or URL as appropriate to the contact type.

Interface: ContactFilter

This interface is a member of [nokia.maps.advsearch](#).

Interface Summary

This interface defines a filter type to narrow down the search scope with contact information.

[For full details, see [nokia.maps.advsearch.ContactFilter](#)]

Table 49: Property Summary

Properties
email: {String} This property holds an email address.
fax: {String} This property holds a fax number.
IM: {String} This property holds an instant messaging account address.
phone: {String} This property holds a phone number.
URL: {String} This property holds the URL of the Web site that belongs to a place.

here

Interface Description

This interface defines a filter type to narrow down the search scope with contact information.

Property Details

email: {String}

This property holds an email address.

fax: {String}

This property holds a fax number.

IM: {String}

This property holds an instant messaging account address.

phone: {String}

This property holds a phone number.

URL: {String}

This property holds the URL of the Web site that belongs to a place.

Interface: Location

This interface is a member of [nokia.maps.advsearch](#).

Interface Summary

This interface defines a location.

[For full details, see [nokia.maps.advsearch.Location](#)]

Table 50: Property Summary

Properties
additionalData: {Object}
This property is a generic key/value container for additional attributes:

here

Properties

address: {*nokia.maps.advsearch.Address*}

This property holds a reference to an optional instance of *nokia.maps.advsearch.Address*.

alternativeLabels: {*nokia.maps.advsearch.AlternativeValue*[]}

This property holds an array of instances of *nokia.maps.advsearch.AlternativeValue*.

displayPosition: {*nokia.maps.geo.Coordinate*}

This property holds a latitude-longitude object indicating where map markers should be placed to indicate the location on the map.

label: {String}

This property holds a descriptive display label for the location in the specified primary language.

locationId: {String}

This property holds a key uniquely identifying a physical location.

locationScore: {Number}

This property holds a value between 0 and 1 which indicates the quality of the underlying location data.

locationType: {String}

This property indicates whether the location is an area or point address.

mapReference: {*nokia.maps.advsearch.MapReference*}

This property holds a reference to an optional map reference object.

mapView: {*nokia.maps.geo.BoundingBox*}

This property holds a bounding box which completely encloses the physical extent of the location.

navigationPositions: {*nokia.maps.geo.Coordinate*[]}

This property holds a latitude-longitude object to be used when defining routing waypoints.

Interface Description

This interface defines a location. The location type refers to a physical location including the physical extent. A location can be referenced either by LRO ID or by specifying the address.

Property Details

additionalData: {Object}

This property is a generic key/value container for additional attributes:

address: {*nokia.maps.advsearch.Address*}

This property holds a reference to an optional instance of *nokia.maps.advsearch.Address*.

here

alternativeLabels: {[nokia.maps.advsearch.AlternativeValue](#)[]}}

This property holds an array of instances of [nokia.maps.advsearch.AlternativeValue](#). Note that the array may be empty.

displayPosition: {[nokia.maps.geo.Coordinate](#)}

This property holds a latitude-longitude object indicating where map markers should be placed to indicate the location on the map.

label: {String}

This property holds a descriptive display label for the location in the specified primary language.

locationId: {String}

This property holds a key uniquely identifying a physical location.

locationScore: {Number}

This property holds a value between 0 and 1 which indicates the quality of the underlying location data.

locationType: {String}

This property indicates whether the location is an area or point address. The possible values are:

- "area"
- "point"
- "line"

mapReference: {[nokia.maps.advsearch.MapReference](#)}

This property holds a reference to an optional map reference object.

mapView: {[nokia.maps.geo.BoundingBox](#)}

here

This property holds a bounding box which completely encloses the physical extent of the location.

navigationPositions: {[nokia.maps.geo.Coordinate](#)[][]}

This property holds a latitude-longitude object to be used when defining routing waypoints.

Interface: LocationFilter

This interface is a member of [nokia.maps.advsearch](#).

Interface Summary

This interface defines a location search results filter type that matches certain address field values or the exact location specified via Location Reference Object id (LRO ID).

[For full details, see [nokia.maps.advsearch.LocationFilter](#)]

Table 51: Property Summary

Properties
city: {String} This property holds the identifier of the administration area at GDF Order8 level (for example US: City, DE: Gemeinde).
country: {String} This property holds the country code (3 bytes, ISO 3166-1-alpha-3).
county: {String} This property holds the identifier of the administration area at GDF Order2 level (for example US: County, DE: Kreis).
district: {String} This property holds the identifier of the administration area at GDF Builtup level (US: n/a, DE: Ortsteil).
houseNumber: {String} This property holds the house number.
locationId: {String} This property holds the Location Reference Object ID, which uniquely identifies a physical location within the LRO system.
postalCode: {String} This property holds the post code (zip in the US).
state: {String} This property holds the identifier of the administration area at GDF Order1 level (for example US: State, DE: Bundesland).
street: {String[]} This property holds the name of the street.

here

Interface Description

This interface defines a location search results filter type that matches certain address field values or the exact location specified via Location Reference Object id (LRO ID).

Property Details

city: {String}

This property holds the identifier of the administration area at GDF Order8 level (for example US: City, DE: Gemeinde). It can also contain an alternative administrative name (for example, "Paris" for Charles-de-Gaulle airport, which is actually located in Roissy-en-France).

country: {String}

This property holds the country code (3 bytes, ISO 3166-1-alpha-3).

county: {String}

This property holds the identifier of the administration area at GDF Order2 level (for example US: County, DE: Kreis).

district: {String}

This property holds the identifier of the administration area at GDF Builtup level (US: n/a, DE: Ortsteil).

houseNumber: {String}

This property holds the house number. Depending on regional characteristics, the property can alternatively contain the house name.

locationId: {String}

This property holds the Location Reference Object ID, which uniquely identifies a physical location within the LRO system.

postalCode: {String}

here

This property holds the post code (zip in the US).

state: {String}

This property holds the identifier of the administration area at GDF Order1 level (for example US: State, DE: Bundesland).

street: {String[]}

This property holds the name of the street. A second street name can be included to search for street intersections.

Interface: LocationMatchQuality

This interface is a member of [nokia.maps.advsearch](#).

Interface Summary

This interface defines a type that provides detailed information about the match quality of a result at attribute level.

[For full details, see [nokia.maps.advsearch.LocationMatchQuality](#)]

Table 52: Property Summary

Properties
city: {Number} This property indicates the match quality of the result with respect to nokia.maps.advsearch.LocationFilter#city.
country: {Number} This property indicates the match quality of the result with respect to nokia.maps.advsearch.LocationFilter#country.
county: {Number} This property indicates the match quality of the result with respect to nokia.maps.advsearch.LocationFilter#county.
district: {Number} This property indicates the match quality of the result with respect to nokia.maps.advsearch.LocationFilter#district.
houseNumber: {Number} This property indicates the match quality of the result with respect to nokia.maps.advsearch.LocationFilter#houseNumber.
postalCode: {Number} This property indicates the match quality of the result with respect to nokia.maps.advsearch.LocationFilter#postalCode.
state: {Number}

here

Properties

This property indicates the match quality of the result with respect to [nokia.maps.advsearch.LocationFilter#state](#).

street: {Number}

This property indicates the match quality of the result with respect to [nokia.maps.advsearch.LocationFilter#street](#).

Interface Description

This interface defines a type that provides detailed information about the match quality of a result at attribute level.

Property Details

city: {Number}

This property indicates the match quality of the result with respect to [nokia.maps.advsearch.LocationFilter#city](#). The match quality is a value between 0.0 and 1.0, 1.0 representing a 100% match.

country: {Number}

This property indicates the match quality of the result with respect to [nokia.maps.advsearch.LocationFilter#country](#). The match quality is a value between 0.0 and 1.0, 1.0 representing a 100% match.

county: {Number}

This property indicates the match quality of the result with respect to [nokia.maps.advsearch.LocationFilter#county](#). The match quality is a value between 0.0 and 1.0, 1.0 representing a 100% match.

district: {Number}

This property indicates the match quality of the result with respect to [nokia.maps.advsearch.LocationFilter#district](#). The match quality is a value between 0.0 and 1.0, 1.0 representing a 100% match.

houseNumber: {Number}

here

This property indicates the match quality of the result with respect to [nokia.maps.advsearch.LocationFilter#houseNumber](#). The match quality is a value between 0.0 and 1.0, 1.0 representing a 100% match.

postalCode: {Number}

This property indicates the match quality of the result with respect to [nokia.maps.advsearch.LocationFilter#postalCode](#). The match quality is a value between 0.0 and 1.0, 1.0 representing a 100% match.

state: {Number}

This property indicates the match quality of the result with respect to [nokia.maps.advsearch.LocationFilter#state](#). The match quality is a value between 0.0 and 1.0, 1.0 representing a 100% match.

street: {Number}

This property indicates the match quality of the result with respect to [nokia.maps.advsearch.LocationFilter#street](#). The match quality is a value between 0.0 and 1.0, 1.0 representing a 100% match.

Class: Manager

This class is a member of [nokia.maps.advsearch](#).

Extends: [nokia.maps.search.Manager](#)

Class Summary

This class provides a client-side interface to search functionality and storage for previous search requests and responses.

[For full details, see [nokia.maps.advsearch.Manager](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.search.Manager](#) :

[gen](#), [locations](#), [maxResults](#), [places](#), [state](#)

here

Table 53: Method Summary

Methods
<code>clear()</code> This method resets the stored search requests and responses.
<code>geocode(SearchRequest)</code> This method performs geocoding by submitting a request to the back-end Search Service.
<code>reverseGeocode(SearchRequest)</code> This method performs a reverse geocoding request based on the given position.

Directly Inherited Methods

Inherited from class [nokia.maps.search.Manager](#) :

[clear](#), [geocode](#), [getErrorCause](#), [reverseGeocode](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This class provides a client-side interface to search functionality and storage for previous search requests and responses. Old search requests and responses can be deleted using the method [nokia.maps.advsearch.Manager#clear](#).

Constructor Details

`nokia.maps.advsearch.Manager()`

This constructor initializes a new empty Manager instance.

Method Details

`clear()`

This method resets the stored search requests and responses.

`geocode(SearchRequest)`

This method performs geocoding by submitting a request to the back-end Search Service.

Parameters:

here

SearchRequest: {[nokia.maps.advsearch.SearchRequest](#)}

A set of parameters for the geocoding request

reverseGeocode(SearchRequest)

This method performs a reverse geocoding request based on the given position. A default radius is applied.

Parameters:

SearchRequest: {[nokia.maps.advsearch.SearchRequest](#)}

A set of parameters for the reverse-geocoding request

Interface: MapReference

This interface is a member of [nokia.maps.advsearch](#).

Interface Summary

This interface defines a map reference to a network link of the location object.

[For full details, see [nokia.maps.advsearch.MapReference](#)]

Table 54: Property Summary

Properties
mapId: {String} This property holds a reference to a specific map.
mapVersion: {String} This property holds the full version identifier of the map.
referenceId: {String} This property holds the id of the given instance of map reference.
sideOfStreet: {String} This property indicates whether the referenced location is on the left or right of the link (if heading from the reference node to the non-reference node).
spot: {Number} This property holds the relative position of the location along the link expressed as the fractional distance from the link's reference-node to the non-reference node.

here

Interface Description

This interface defines a map reference to a network link of the location object.

Property Details

mapId: {String}

This property holds a reference to a specific map.

mapVersion: {String}

This property holds the full version identifier of the map.

referenceId: {String}

This property holds the id of the given instance of map reference.

sideOfStreet: {String}

This property indicates whether the referenced location is on the left or right of the link (if heading from the reference node to the non-reference node). The possible values are:

- "left"
- "right"
- "neither"

spot: {Number}

This property holds the relative position of the location along the link expressed as the fractional distance from the link's reference-node to the non-reference node. The value of the property is in the range between 0 and 1. This property is relevant only if a link is referenced.

Interface: Place

This interface is a member of [*nokia.maps.advsearch*](#).

Interface Summary

This interface represents a place and its properties.

here

[For full details, see [nokia.maps.advsearch.Place](#)]

Table 55: Property Summary

Properties
<code>additionalData</code>: {Object} This property is a generic key/value container to keep additional attributes.
<code>alternativeNames</code>: {nokia.maps.advsearch.AlternativeValue[]} This property holds an array of <code>AlternativeValue</code> objects.
<code>categories</code>: {nokia.maps.advsearch.Category[]} This property holds an array of <code>Category</code> objects.
<code>contacts</code>: {nokia.maps.advsearch.Contact} This property holds a reference to an optional <code>Contact</code> object.
<code>locations</code>: {nokia.maps.advsearch.Location[]} This property holds an array of <code>Location</code> objects.
<code>name</code>: {String} This property holds a descriptive name of the place in the specified primary language for display purposes.
<code>placeId</code>: {String} This property holds a unique id of the place.
<code>placeScore</code>: {Number} This property holds a value which indicates the quality of the underlying place data.
<code>primaryEmail</code>: {String} This property holds the primary email address of the place.
<code>primaryFax</code>: {String} This property holds the primary fax number of the place.
<code>primaryIM</code>: {String} This property holds the primary instant messaging account of the place.
<code>primaryPhone</code>: {String} This property holds the primary phone number of the place.
<code>primaryURL</code>: {String} This property holds the primary URL of the place.
<code>suppliers</code>: {nokia.maps.advsearch.Supplier[]} This property holds an array of <code>Supplier</code> objects.

here

Interface Description

This interface represents a place and its properties. A place is an entity that encapsulates information about objects, services, etc., that can be found at a specific location. A place can represent a single point defined its latitude and longitude or it may be an area or feature of the landscape identified as a Point of Interest (POI). A place often has information such as contact details and media (images and reviews) associated with it.

Property Details

additionalData: {Object}

This property is a generic key/value container to keep additional attributes.

alternativeNames: {*nokia.maps.advsearch.AlternativeValue*[][]}

This property holds an array of *AlternativeValue* objects. Note that the array can be empty.

categories: {*nokia.maps.advsearch.Category*[][]}

This property holds an array of *Category* objects. Note that the array can be empty.

contacts: {*nokia.maps.advsearch.Contact*}

This property holds a reference to an optional *Contact* object.

locations: {*nokia.maps.advsearch.Location*[][]}

This property holds an array of *Location* objects. It contains at least one element.

name: {String}

This property holds a descriptive name of the place in the specified primary language for display purposes.

placeId: {String}

This property holds a unique id of the place.

here

placeScore: {Number}

This property holds a value which indicates the quality of the underlying place data.

primaryEmail: {String}

This property holds the primary email address of the place.

primaryFax: {String}

This property holds the primary fax number of the place.

primaryIM: {String}

This property holds the primary instant messaging account of the place.

primaryPhone: {String}

This property holds the primary phone number of the place.

primaryURL: {String}

This property holds the primary URL of the place.

suppliers: {[nokia.maps.advsearch.Supplier](#)[][]}

This property holds an array of [Supplier](#) objects. It contains at least one element.

Interface: PlaceFilter

This interface is a member of [nokia.maps.advsearch](#).

Interface Summary

This interface defines a filter type to match search results to certain place values.

[For full details, see [nokia.maps.advsearch.PlaceFilter](#)]

here

Table 56: Property Summary

Properties
<code>categoryFilter</code> : { <i>nokia.maps.advsearch.CategoryFilter</i> []} This property holds an array of <code>CategoryFilter</code> objects.
<code>contactFilter</code> : { <i>nokia.maps.advsearch.ContactFilter</i> } This property holds a reference to an optional <code>ContactFilter</code> object.

Interface Description

This interface defines a filter type to match search results to certain place values.

Property Details

`categoryFilter`: {[*nokia.maps.advsearch.CategoryFilter*](#)[]}

This property holds an array of `CategoryFilter` objects. Note that the array can be empty.

`contactFilter`: {[*nokia.maps.advsearch.ContactFilter*](#)}

This property holds a reference to an optional `ContactFilter` object.

Interface: ReverseGeocodeSettings

This interface is a member of [*nokia.maps.advsearch*](#).

Interface Summary

This interface represents settings which apply to reverse geocoding requests.

[For full details, see [*nokia.maps.advsearch.ReverseGeocodeSettings*](#)]

Table 57: Property Summary

Properties
<code>mode</code> : {String} This field specifies the mode for reverse geocoding requests.

Interface Description

This interface represents settings which apply to reverse geocoding requests.

here

Property Details

mode: {String}

This field specifies the mode for reverse geocoding requests. The possible values are:

- "retrieveAddresses" - indicates that addresses near the specified terminal position are to be retrieved
- "retrieveAreas" - indicates that the area around the specified terminal position is to be retrieved

Interface: SearchRepresentationOptions

This interface is a member of [nokia.maps.advsearch](#).

Interface Summary

This interface represents a set of parameters designed to control the representation of search results.

[For full details, see [nokia.maps.advsearch.SearchRepresentationOptions](#)]

Table 58: Property Summary

Properties
addressAttributes: {String[]} This property holds an array of AddressAttributes objects.
language: {String} This property identifies the preferred languages in which internationalized text attributes should be returned in the response.
locationAttributes: {String[]} This property holds an array of LocationAttributes objects.
maxResults: {Number} This property holds a value indicating the maximum number of items to be included in the response structure.
pageInformation: {String} This property identifies the page to be returned in a paging scenario.
placeAttributes: {String[]} This property holds an array of PlaceAttributes objects.
searchResponseAttributes: {String[]} This property holds an array of values that determine the attributes to be returned in the search response.

here

Properties

`strictLanguageMode`: {Boolean}

This property holds a flag indicating whether the strict language mode applies or not.

Interface Description

This interface represents a set of parameters designed to control the representation of search results. The parameters include the language of text attributes and the detail level of information to be returned.

Property Details

`addressAttributes`: {String[]}

This property holds an array of AddressAttributes objects. The property is optional, therefore the array may be empty.

The possible values are:

- "ctr" - "country"
- "sta" - "state"
- "cty" - "county"
- "cit" - "city"
- "dis" - "district"
- "str" - "street"
- "hnr" - "houseNumber"
- "pst" - "postalCode"
- "sui" - "suite"
- "flr" - "floor"
- "add" - "additionalAttributes"
- "alt" - "alternativeAttributes"
- "none" - ("none"), indicates that none of the optional attributes are to be provided in the response; "none" clears the default set of attributes and allows the user to add exactly those attributes he/she requires
- "all" - ("all"), indicates that all attributes configured by this enumeration type are to be provided in the response; this value can be used when the user wishes to remove specific attributes from a predefined list (the reference set is the list of all attributes)

here

language: {String}

This property identifies the preferred languages in which internationalized text attributes should be returned in the response. The list is assumed to be in descending order. The filter is specified according to RFC 4647.

locationAttributes: {String[]}

This property holds an array of `LocationAttributes` objects. The property is optional and therefore the array may be empty.

The possible values are:

- "ar" - "address"
- "mr" - "mapReference"
- "mv" - "mapView"
- "ad" - "additionalData"
- "al" - "alternativeLabels"
- "none" - ("none"), indicates that none of the optional attributes are to be provided in the response; "none" clears the default set of attributes and allows the user to add exactly those attributes he/she requires
- "all" ("all"), indicates that all attributes configured by this enumeration type are to be provided in the response; this value can be used when the user wishes to remove specific attributes from a predefined list (the reference set is the list of all attributes)

maxResults: {Number}

This property holds a value indicating the maximum number of items to be included in the response structure. If this property is specified, paging is activated and the results are returned in separate pages. Each response contains a handle to the next page.

pageInformation: {String}

This property identifies the page to be returned in a paging scenario. The property is taken into account only if `MaxResults` has been specified in a previous request.

placeAttributes: {String[]}

here

This property holds an array of PlaceAttributes objects. The property is optional and therefore the array may be empty.

The possible values are:

- "ca" - "categories"
- "lo" - "location"
- "sp" - "supplier"
- "co" - "contact"
- "ad" - "additionalData"
- "an" - "alternativeNames"
- "none" - ("none"), indicates that none of the optional attributes are to be provided in the response; "none" clears the default set of attributes and allows the user to add exactly those attributes he/she requires
- "all" ("all"), indicates that all attributes configured by this enumeration type are to be provided in the response; this value can be used when the user wishes to remove specific attributes from a predefined list (the reference set is the list of all attributes)

searchResponseAttributes: {String[]}

This property holds an array of values that determine the attributes to be returned in the search response. The property is optional. If it is not specified, all search response attributes apart from the performed search association are included by default (the response contains all values besides apart from PerformedSearch and MatchQuality).

The possible values are:

- "ps" ("performedSearch") - indicates if the performed search information should be returned
- "la" ("label") - indicates if a label for the SearchResponseView is to be returned
- "dm" ("didYouMeanSuggestion") - indicates if didYouMeanSuggestions is to be returned
- "mq" ("matchQuality") - indicates if the match quality on attribute level is to be returned
- "mt" ("matchType") - indicates whether the location could be determined exactly or if an interpolation was applied
- "none" ("none") - indicates that none of the optional attributes are to be provided in the response; "none" clears the default set of attributes and allows the user to add exactly those attributes he/she requires
- "all" ("all") - indicates that all attributes configured by this enumeration type are to be provided in the response; this value can be used when the user wishes to remove specific attributes from a predefined list (the reference set is the list of all attributes)

here

strictLanguageMode: {Boolean}

This property holds a flag indicating whether the strict language mode applies or not. If its value is `false`, the back end chooses the best available attribute value reflecting the language preferences and returns them directly in `Address`, `Place`, `Location`, and `Category`. In this case, the language information is lost. If the value of the property is `true`, the service includes only the attribute value directly in `Address`, `Place`, `Location`, and `Category` if a value is available in the first language specified by the property [nokia.maps.advsearch.SearchRepresentationOptions#language](#). Values in alternative languages are returned in the associated instances of [nokia.maps.advsearch.AlternativeValue](#).

Interface: SearchRequest

This interface is a member of [nokia.maps.advsearch](#).

Interface Summary

This interface defines a search request with parameters supported by all search types.

[For full details, see [nokia.maps.advsearch.SearchRequest](#)]

Table 59: Property Summary

Properties
additionalData: {Object} This property is a generic container for additional information to be attached to the request.
locationFilter: {nokia.maps.search.LocationFilter} This property holds a reference to an optional <code>LocationFilter</code> object.
metaInfo: {nokia.maps.advsearch.SearchRequestMetaInfo} This property holds a reference to an optional <code>RequestMetaInfo</code> object.
placeFilter: {nokia.maps.advsearch.PlaceFilter} This property holds a reference to an optional <code>PlaceFilter</code> object.
representationOptions: {nokia.maps.advsearch.SearchRepresentationOptions} This property holds a reference to an optional <code>RepresentationOptions</code> object.
reverseGeocodeSettings: {nokia.maps.advsearch.ReverseGeocodeSettings} This property holds a reference to an optional <code>ReverseGeocodeSettings</code> object.
searchText: {String} This property holds free text input which is dynamically matched against different fields.

here

Properties

spatialFilter: {[nokia.maps.geo.BoundingBox](#) | [nokia.maps.geo.Corridor](#) | [nokia.maps.geo.Proximity](#)}

This property holds a reference to an optional object representing a spatial filter, which restricts search results to places/locations that are included within a certain geospatial area.

Interface Description

This interface defines a search request with parameters supported by all search types.

Property Details

additionalData: {Object}

This property is a generic container for additional information to be attached to the request.

locationFilter: {[nokia.maps.search.LocationFilter](#)}

This property holds a reference to an optional LocationFilter object. The response contains only those locations that match the attributes of this object.

metaInfo: {[nokia.maps.advsearch.SearchRequestMetaInfo](#)}

This property holds a reference to an optional RequestMetaInfo object.

placeFilter: {[nokia.maps.advsearch.PlaceFilter](#)}

This property holds a reference to an optional PlaceFilter object. The response contains only those places that have match the attributes of this object.

representationOptions: {[nokia.maps.advsearch.SearchRepresentationOptions](#)}

This property holds a reference to an optional RepresentationOptions object.

reverseGeocodeSettings: {[nokia.maps.advsearch.ReverseGeocodeSettings](#)}

This property holds a reference to an optional ReverseGeocodeSettings object.

searchText: {String}

here

This property holds free text input which is dynamically matched against different fields.

spatialFilter: {[nokia.maps.geo.BoundingBox](#) | [nokia.maps.geo.Corridor](#) | [nokia.maps.geo.Proximity](#)}

This property holds a reference to an optional object representing a spatial filter, which restricts search results to places/locations that are included within a certain geospatial area. Instances of `Islocation.geo.BoundingBox`, [nokia.maps.geo.Corridor](#), and [nokia.maps.geo.Proximity](#) are allowed as spatial filters.

Interface: SearchRequestMetaInfo

This interface is a member of [nokia.maps.advsearch](#).

Interface Summary

This interface defines parameters not directly related to search functionality.

[For full details, see [nokia.maps.advsearch.SearchRequestMetaInfo](#)]

Table 60: Property Summary

Properties
requestId: {String} This property holds an arbitrary value to be mirrored in the response structure.
verboseMode: {Number} This property holds a numeric value indicating the required verbosity level in the response structure.

Interface Description

This interface defines parameters not directly related to search functionality.

Property Details

requestId: {String}

This property holds an arbitrary value to be mirrored in the response structure. The value can be used to trace back requests.

verboseMode: {Number}

here

This property holds a numeric value indicating the required verbosity level in the response structure. The permitted values are in the range between 0 and 5.

Interface: SearchResponse

This interface is a member of [nokia.maps.advsearch](#).

Interface Summary

This interface represents the top-level element returned by the search service.

[For full details, see [nokia.maps.advsearch.SearchResponse](#)]

Table 61: Property Summary

Properties
metaInfo: { nokia.maps.advsearch.SearchResponseMetaInfo } This property holds a reference to an optional response meta info object.
view: { nokia.maps.advsearch.SearchResponseView []} This property holds an array of search response view objects.

Interface Description

This interface represents the top-level element returned by the search service. It contains the search results along with meta information, search paths, and scores.

Property Details

metaInfo: {[nokia.maps.advsearch.SearchResponseMetaInfo](#)}

This property holds a reference to an optional response meta info object.

view: {[nokia.maps.advsearch.SearchResponseView](#)[]}
 This property holds an array of search response view objects. It contains at least one element.

Interface: SearchResponseMetaInfo

This interface is a member of [nokia.maps.advsearch](#).

here

Interface Summary

This interface encapsulates information not related to a search type.

[For full details, see [nokia.maps.advsearch.SearchResponseMetaInfo](#)]

Table 62: Property Summary

Properties
<code>additionalData</code>: {Object} This property holds an object containing additional data.
<code>nextPageInformation</code>: {String} This property holds a key that can be used in subsequent requests to acquire the next n results.
<code>previousPageInformation</code>: {String} This property holds a key that can be used in subsequent requests to acquire the previous n results.
<code>requestId</code>: {String} This property holds the mirrored request id, which is a value from the request structure.
<code>timestamp</code>: {String} This property holds the time stamp indicating when the search was performed.

Interface Description

This interface encapsulates information not related to a search type.

Property Details

`additionalData`: {Object}

This property holds an object containing additional data. The contents depend on the user's requirements.

`nextPageInformation`: {String}

This property holds a key that can be used in subsequent requests to acquire the next n results. This element is only provided if paging has been activated in the request.

`previousPageInformation`: {String}

This property holds a key that can be used in subsequent requests to acquire the previous n results. This element is only provided if paging has been activated in the request.

here

requestId: {String}

This property holds the mirrored request id, which is a value from the request structure. The property can be used to trace back requests.

timestamp: {String}

This property holds the time stamp indicating when the search was performed.

Interface: SearchResponseView

This interface is a member of [nokia.maps.advsearch](#).

Interface Summary

This interface represents a specialized view of the search response to be returned to the requesting application.

[For full details, see [nokia.maps.advsearch.SearchResponseView](#)]

Table 63: Property Summary

Properties
label: {String} This property holds a descriptive label of the view for display purposes.
viewId: {Number} This property holds the identifier of the given view.

Interface Description

This interface represents a specialized view of the search response to be returned to the requesting application. It reflects (attempts to match) the assumed intention behind the user's free-text search.

Property Details

label: {String}

This property holds a descriptive label of the view for display purposes. The label is in the configured primary language.

here

viewId: {Number}

This property holds the identifier of the given view.

Interface: SearchResult

This interface is a member of [nokia.maps.advsearch](#).

Interface Summary

This interface represents a search result item.

[For full details, see [nokia.maps.advsearch.SearchResult](#)]

Table 64: Property Summary

Properties
additionalData: {Object} This property holds an object containing additional data.
direction: {Number} This property holds the direction of the location object as seen from the specified client position measured clockwise in degrees starting from 0 (North).
distance: {Number} This property holds the distance between the identified location object and the specified client position in meters (only provided if a nokia.maps.geo.Proximity is used as a spatial filter in nokia.maps.advsearch.SearchRequest).
location: {nokia.maps.advsearch.Location} This property holds a mandatory reference to an instance of nokia.maps.advsearch.Location.
matchLevel: {String} This property holds the name of the most detailed address field which could be matched (only relevant for geocoding results).
matchQuality: {nokia.maps.advsearch.LocationMatchQuality} This property holds detailed information about the match quality on attribute level.
matchType: {String} This property holds a value indicating whether the result was matched to a point address or whether the match is interpolated.
place: {nokia.maps.advsearch.Place} This property holds a mandatory reference to an instance of nokia.maps.advsearch.Place.
relevance: {Number} This property indicates the relevance of the results found; the higher the score the more relevant the alternative.

here

Interface Description

This interface represents a search result item.

Property Details

additionalData: {Object}

This property holds an object containing additional data. The contents depend on the user's requirements.

direction: {Number}

This property holds the direction of the location object as seen from the specified client position measured clockwise in degrees starting from 0 (North).

distance: {Number}

This property holds the distance between the identified location object and the specified client position in meters (only provided if a [nokia.maps.geo.Proximity](#) is used as a spatial filter in [nokia.maps.advsearch.SearchRequest](#)).

location: {[nokia.maps.advsearch.Location](#)}

This property holds a mandatory reference to an instance of [nokia.maps.advsearch.Location](#).

matchLevel: {String}

This property holds the name of the most detailed address field which could be matched (only relevant for geocoding results).

The possible values are:

- "ctr" - "country"
- "sta" - "state"
- "cty" - "county"
- "cit" - "city"
- "dis" - "district"
- "str" - "street"
- "hnr" - "houseNumber"

here

- "pst" - "postalCode"
- "sui" - "suite"
- "flr" - "floor"
- "add" - "additionalAttributes"
- "alt" - "alternativeAttributes"
- "none" - ("none"), indicates that none of the optional attributes are to be provided in the response; "none" clears the default set of attributes and allows the user to add exactly those attributes he/she requires
- "all" - ("all") indicates that all attributes configured by this enumeration type are to be provided in the response; this value can be used when the user wishes to remove specific attributes from a predefined list (the reference set is the list of all attributes)

matchQuality: {[*nokia.maps.advsearch.LocationMatchQuality*](#)}

This property holds detailed information about the match quality on attribute level.

matchType: {String}

This property holds a value indicating whether the result was matched to a point address or whether the match is interpolated. The possible values are:

- "pointAddress"
- "interpolated"

place: {[*nokia.maps.advsearch.Place*](#)}

This property holds a mandatory reference to an instance of [*nokia.maps.advsearch.Place*](#).

relevance: {Number}

This property indicates the relevance of the results found; the higher the score the more relevant the alternative.

Interface: SearchResultsView

This interface is a member of [*nokia.maps.advsearch*](#).

here

Interface Summary

This interface defines a specialized view of the search results to be returned to the requesting application.

[For full details, see [nokia.maps.advsearch.SearchResultsView](#)]

Table 65: Property Summary

Properties
didYouMeanSuggestion: {String[]} This property holds a list of strings that contain "did you mean" suggestions.
performedSearch: {nokia.maps.advsearch.SearchRequest} This property holds a reference to an instance of nokia.maps.advsearch.SearchRequest.
results: {nokia.maps.advsearch.SearchResult[]} This property holds an array of instances of nokia.maps.advsearch.SearchResult.

Interface Description

This interface defines a specialized view of the search results to be returned to the requesting application.

Property Details

[didYouMeanSuggestion](#): {String[]}

This property holds a list of strings that contain "did you mean" suggestions.

[performedSearch](#): {[nokia.maps.advsearch.SearchRequest](#)}

This property holds a reference to an instance of [nokia.maps.advsearch.SearchRequest](#). The use of the property is optional.

[results](#): {[nokia.maps.advsearch.SearchResult](#)[]}

This property holds an array of instances of [nokia.maps.advsearch.SearchResult](#). The use of the property is optional, therefore the array can be empty.

Interface: SearchSuggestionsView

This interface is a member of [nokia.maps.advsearch](#).

here

Interface Summary

This interface defines a specialized view of input search suggestions to be returned to the application.

[For full details, see [nokia.maps.advsearch.SearchSuggestionsView](#)]

Table 66: Property Summary

Properties
<code>typeSuggestion</code> : {String} This property holds the suggestion text that can be used to autocomplete the text the user is currently typing.

Interface Description

This interface defines a specialized view of input search suggestions to be returned to the application.

Property Details

`typeSuggestion`: {String}

This property holds the suggestion text that can be used to autocomplete the text the user is currently typing.

Interface: Supplier

This interface is a member of [nokia.maps.advsearch](#).

Interface Summary

This interface defines a supplier of place information.

[For full details, see [nokia.maps.advsearch.Supplier](#)]

Table 67: Property Summary

Properties
<code>name</code> : {String} This property holds a descriptive name of the data provider for display purposes.
<code>supplierId</code> : {String} This property holds a unique identifier of the place data provider.

here

Interface Description

This interface defines a supplier of place information. Provider identification for place information.

Property Details

name: {String}

This property holds a descriptive name of the data provider for display purposes.

supplierId: {String}

This property holds a unique identifier of the place data provider.

Namespace: clustering

This namespace is a member of [nokia.maps](#).

Namespace Summary

This namespace contains resources that allow the API user to cluster markers.

Namespace Description

This namespace contains resources that allow the API user to cluster markers. So areas with high density of markers can be easily displayed on the map. As a result objects are grouped into "islands", aka clusters. From that cluster only one single spatial object is shown as a presentation of that cluster. Namespace also provides some useful classes by using which it's very easy to customize presentation of the cluster.

Class: Cluster

This class is a member of [nokia.maps.clustering](#).

Class Summary

This class represents a cluster of markers on the map.

[For full details, see [nokia.maps.clustering.Cluster](#)]

here

Table 68: Method Summary

Methods
<i>getBounds</i> (dataPoints) : {<i>nokia.maps.geo.BoundingBox</i>} This method retrieves an instance of <i>nokia.maps.geo.BoundingBox</i> representing the smallest bounding box that encloses all the data points in the cluster.
<i>getIcon</i> (theme, resetTheMarker) This method retrieves the icon that represents the given cluster.
<i>getPoints</i> () : {<i>nokia.maps.clustering.IClusterPoint</i>[]} This method retrieves the data points that make up the given cluster.
<i>getSize</i> () : {Number} This method retrieves the count of the data points in the given cluster.

Class Description

This class represents a marker cluster on the map. It contains references to all the data points that make up the cluster and provides methods to manage them.

Constructor Details

nokia.maps.clustering.Cluster(point, opt)

This method creates an instance of Cluster.

Parameters:

- point: {*nokia.maps.clustering.IClusterPoint*} [optional]
Seed (or the first data point) of the cluster; further data points can be added to the cluster.
- opt: {*nokia.maps.clustering.IGridOptions*} [optional]
Configuration options for cluster in case if grid based clustering is used

Method Details

getBounds(dataPoints): {*nokia.maps.geo.BoundingBox*}

This method retrieves an instance of *nokia.maps.geo.BoundingBox* representing the smallest bounding box that encloses all the data points in the cluster.

Parameters:

here

dataPoints: {*nokia.maps.clustering.IClusterPoint*[]}

A collection of objects representing the data points in the cluster

Returns:

{*nokia.maps.geo.BoundingBox*}

The smallest bounding box that encloses all the data points inside the cluster

getIcon(theme, resetTheMarker)

This method retrieves the icon that represents the given cluster.

Parameters:

theme: {*nokia.maps.clustering.ITheme*}

The theme to use to representing the cluster on the map

resetTheMarker: {Boolean} [optional]

A flag indicating whether the marker icon is to be updated (true) or whether the cached icon is to be retrieved (false; this argument can be used, for example, when modifying theme of existing clusters

getPoints(): {*nokia.maps.clustering.IClusterPoint*[]}

This method retrieves the data points that make up the given cluster.

Returns:

{*nokia.maps.clustering.IClusterPoint*[]}

An array of data points that make up the given cluster

getSize(): {Number}

This method retrieves the count of the data points in the given cluster.

Returns:

here

{Number} The number of data points in the given cluster

Class: ClusterProvider

This class is a member of [nokia.maps.clustering](#).

Extends: [nokia.maps.map.provider.Provider](#), [nokia.maps.util.OObject](#)

Class Summary

This class works on a set of data points (locations) and organizes them into clusters to optimize their display on the map.

[For full details, see [nokia.maps.clustering.ClusterProvider](#)]

Table 69: Property Summary

Properties
static STATE_CLUSTERED : {String} This constant indicates a state which an instance of ClusterProvider enters when it has completed clustering and rendering of the markers on the map can begin (is about to begin).
static STATE_INITIAL : {String} This constant identifies the initial state of an instance of this class after the instance has been created, or after invocation of nokia.maps.clustering.ClusterProvider#clean and nokia.maps.clustering.ClusterProvider#invalidate methods.
static STATE_READY : {String} This constant indicates a state which an instance of ClusterProvider enters when it has determined clusters and noise points and their representation for display on the map.
static STATE_STARTED : {String} This constant indicates a state which an instance of ClusterProvider enters when clustering has started.
static STRATEGY_DENSITY_BASED : {Number} This constant indicates that density based clustering strategy should be used.
static STRATEGY_GRID_BASED : {Number} This constant indicates that grid based clustering strategy should be used.

Directly Inherited Properties

Inherited from class [nokia.maps.map.provider.Provider](#) :

[description](#), [getInvalidationMark](#), [id](#), [label](#), [max](#), [min](#)

here

Table 70: Method Summary

Methods
<i>add</i> (dataPoint) This method adds a data point to the list of data points to be considered for clustering.
<i>addAll</i> (dataPoints, callback) This method adds a list of data points to the list of objects for clustering.
<i>clean</i> () This method removes clusters and noise points from display.
<i>cluster</i> () This method starts the clustering process.
<i>destroy</i> () This method cleans up all internal objects and prepares the given instance of this class to be destroyed.
<i>getContainer</i> () : {<i>nokia.maps.map.Container</i>} Returns instance of <i>nokia.maps.map.Container</i> which holds presentations for all clusters and noise points.
<i>getDataLength</i> () : {Number} This method retrieves a numeric value indicating the number of data points in the data set available to the given instance of the class.
<i>getEps</i> () : {Number} This method retrieves the current epsilon value, which represents the area within which data points are considered for clustering.
<i>getMinPts</i> () : {Number} This method retrieves a value indicating the smallest number of points within the epsilon "radius" that are required to form a cluster.
<i>invalidate</i> () This method invalidates the current results of clustering and sets the state to <i>nokia.maps.clustering.ClusterProvider.STATE_INITIAL</i>.
<i>numberOfClusters</i> () : {Number} This method retrieves the number of clusters in the data set available to the given instance of the class.
<i>numberOfNoisePoints</i> () : {Number} This method retrieves a numeric value indicating the number of noise points in the data set available to the given instance of this class.
<i>remove</i> (dataPoint) This method removes a data point from the list of objects to be considered for clustering.
<i>setEps</i> (epsilon) This method sets a new epsilon value (in pixels), which represents the area within which data points are considered for clustering.

here

Methods

setMinPts (minPts)

This method sets a new value indicating the smallest number of points within the epsilon "radius" that are required to form a cluster.

setTheme (theme)

This method sets the theme to use while rendering clusters on the map.

Directly Inherited Methods

Inherited from class *nokia.maps.map.provider.Provider* :

getCopyrights, *providesLevel*, *shutdown*, *update*

Inherited from class *nokia.maps.util.EventTarget* :

addListener, *dispatch*, *removeListener*

Inherited from class *nokia.maps.util.OObject* :

addObserver, *get*, *remove*, *removeObserver*, *set*

Event Summary

Directly Inherited Events

Inherited from class *nokia.maps.map.provider.Provider* :

response, *update*

Class Description

Class is a implementation of map Provider and visualize a given data as cluster.

Constructor Details

nokia.maps.clustering.ClusterProvider(display, options)

This method creates new instance of ClusterProvider.

Parameters:

display: {*nokia.maps.map.Display*}

Display The object on which markers are to be drawn

options: {*nokia.maps.clustering.ClusterProvider.Options*} [optional]

here

An object specifying the configuration settings for the clustering provider instance

Property Details

`static STATE_CLUSTERED: {String}`

This constant indicates a state which an instance of `ClusterProvider` enters when it has completed clustering and rendering of the markers on the map can begin (is about to begin). Please note that, because clustering may be repeated on every zoom level change, this state might be set multiple times.

`static STATE_INITIAL: {String}`

This constant identifies the initial state of an instance of this class after the instance has been created, or after invocation of [nokia.maps.clustering.ClusterProvider#clean](#) and [nokia.maps.clustering.ClusterProvider#invalidate](#) methods.

`static STATE_READY: {String}`

This constant indicates a state which an instance of `ClusterProvider` enters when it has determined clusters and noise points and their representation for display on the map. Note that when this state is set, clusters and noise points may not be visible on the map yet. The rendering engine needs some additional time to draw them on the map.

`static STATE_STARTED: {String}`

This constant indicates a state which an instance of `ClusterProvider` enters when clustering has started. Please note that, because clustering may be repeated on every zoom level change, this state might be set multiple times.

`static STRATEGY_DENSITY_BASED: {Number}`

This constant indicates that density based clustering strategy should be used.

`static STRATEGY_GRID_BASED: {Number}`

This constant indicates that grid based clustering strategy should be used.

here

Method Details

add(dataPoint)

This method adds a data point to the list of data points to be considered for clustering. Note that the method does not perform input validation to boost performance, therefore the caller must ensure that the object passed to the method contains the properties "latitude" and "longitude".

Parameters:

dataPoint: {[nokia.maps.clustering.IClusterPoint](#)}

An object containing the latitude and longitude of a location; it may additionally contain the property "value" if you wish to position the cluster marker according to the "weight" center of the cluster (see also [nokia.maps.clustering.IClusterPoint](#)).

addAll(dataPoints, callback)

This method adds a list of data points to the list of objects for clustering.

Note that to ensure optimal performance, it is best to invoke the method [nokia.maps.clustering.ClusterProvider#cluster](#) after all the data have been added. Otherwise, as in database systems, addition of new data points means that indices may have to be modified, which incurs a performance penalty.

Parameters:

dataPoints: {[nokia.maps.clustering.IClusterPoint](#)[]}

A collection of data points represented by objects containing latitude and longitude; each point may additionally contain the property "value" if you wish to position the cluster marker according to the "weight" center of the cluster (see also [nokia.maps.clustering.IClusterPoint](#)).

callback: {Function} [optional]

A callback function to be invoked when all data points (see dataPoints above) have been added to the list of items to be considered for clustering

clean()

here

This method removes clusters and noise points from display. It removes all the data points and sets the class state to `nokia.maps.clustering.ClusterProvider.STATE_INITIAL`.

cluster()

This method starts the clustering process. Note that the process is asynchronous, therefore to receive notification of its completion, you need to attach an observer to the instance of this class, using the method [nokia.maps.util.OObject#addObserver](#). The observer function must be triggered when the `ClusterProvider` state changes to `nokia.maps.clustering.ClusterProvider.STATE_CLUSTERED`.

```
var ClusterProvider = nokia.maps.clustering.ClusterProvider;
clusterProvider = new ClusterProvider(map, {
  dataPoints: [PASS YOUR DATA POINTS HERE]
});
clusterProvider.addObserver("state", function (obj, key, state, oldValue) {
  if (state == ClusterProvider.STATE_CLUSTERED) {
    console.log("Clustering is done!");
  }
});
clusterProvider.cluster();
```

destroy()

This method cleans up all internal objects and prepares the given instance of this class to be destroyed.

getContainer(): {[nokia.maps.map.Container](#)}

Returns instance of [nokia.maps.map.Container](#) which holds presentations for all clusters and noise points. You can use this method when planning to do for example an event delegation or for attaching common event handlers.

Returns:

{[nokia.maps.map.Container](#)}

A value representing the number of data points

getLength(): {Number}

here

This method retrieves a numeric value indicating the number of data points in the data set available to the given instance of the class.

Returns:

{Number} A value representing the number of data points

getEps(): {Number}

This method retrieves the current epsilon value, which represents the area within which data points are considered for clustering.

Returns:

{Number} The current epsilon value

getMinPts(): {Number}

This method retrieves a value indicating the smallest number of points within the epsilon "radius" that are required to form a cluster.

Returns:

{Number} The current value of minimum points

invalidate()

This method invalidates the current results of clustering and sets the state to `nokia.maps.clustering.ClusterProvider.STATE_INITIAL`. Note that the method does not remove objects from the display: in order to do that, please use the [nokia.maps.clustering.ClusterProvider#clean](#).

numberOfClusters(): {Number}

This method retrieves the number of clusters in the data set available to the given instance of the class.

Returns:

{Number} A value representing the number of clusters

here

Throws:

A run-time error if [nokia.maps.clustering.ClusterProvider#cluster](#) method has not been called (no clusters have been determined).

numberOfNoisePoints(): {Number}

This method retrieves a numeric value indicating the number of noise points in the data set available to the given instance of this class.

Returns:

{Number} A value indicating the number of noise points

Throws:

A run-time error if [nokia.maps.clustering.ClusterProvider#cluster](#) method has not been called (no noise points have been determined)

remove(dataPoint)

This method removes a data point from the list of objects to be considered for clustering.

Parameters:

dataPoint: {[nokia.maps.clustering.IClusterProvider](#)}

An object containing latitude and longitude (a data point) to be removed from the list of objects to be considered for clustering

Throws:

If indexing is enabled, the method throws a run-time error exception as described for [nokia.maps.clustering.Index#remove](#)

setEps(epsilon)

This method sets a new epsilon value (in pixels), which represents the area within which data points are considered for clustering.

Note that clusters should be recalculated when the epsilon value changes. Please call the method [nokia.maps.clustering.ClusterProvider#cluster](#) explicitly to force a recalculation.

here

Parameters:

epsilon: {Number}

A new epsilon value; it should be a value no smaller than 10

setMinPts(minPts)

This method sets a new value indicating the smallest number of points within the epsilon "radius" that are required to form a cluster. Note that clusters should be recalculated whenever this method has been called. Please call the method [nokia.maps.clustering.ClusterProvider#cluster](#) explicitly to force a recalculation.

Parameters:

minPts: {Number}

A value indicating the largest number of data points within the epsilon "radius" that can exist as individual noise points (that are not subject to clustering); the value must be no smaller than 1

setTheme(theme)

This method sets the theme to use while rendering clusters on the map. Please note, that theme modification triggers the `cluster()` method.

Parameters:

theme: {[nokia.maps.clustering.ITheme](#)}

An object representing the new theme to use

Throws:

A run-time error if the theme object does not conform to the interface [nokia.maps.clustering.ITheme](#)

Interface: Options

This interface is a member of [nokia.maps.clustering.ClusterProvider](#).

here

Interface Summary

This interface defines the configuration options for [nokia.maps.clustering.ClusterProvider](#)

[For full details, see [nokia.maps.clustering.ClusterProvider.Options](#)]

Table 71: Property Summary

Properties
dataPoints: {nokia.maps.clustering.IClusterPoint[]} This property holds a collection of objects representing data points.
eps: {Number} This property holds the "radius" (in pixels) within which data points are considered for clustering.
index: {nokia.maps.clustering.Index} This property holds an indexing data structure to use along with the clustering data.
max: {Number} This property holds a value of the maximum zoom level at which to perform clustering.
min: {Number} This property holds a value of the minimum zoom level at which to perform clustering.
minPts: {Number} This property holds the smallest number of points that must be present within the eps distance of an arbitrary point to form a cluster.
strategy: {Number} This property specifies strategy to use during clustering.
theme: {nokia.maps.clustering.ITheme} This property holds an object that represents the display theme for clusters and noise points.

Interface Description

This interface defines the configuration options for [nokia.maps.clustering.ClusterProvider](#)

Property Details

dataPoints: {[nokia.maps.clustering.IClusterPoint](#)[]}

This property holds a collection of objects representing data points. Each object contains the latitude and longitude of a location, but may also include the property value for weight-based positioning of the cluster markers - see also [nokia.maps.clustering.IClusterPoint](#) for further information.

here

eps: {Number}

This property holds the "radius" (in pixels) within which data points are considered for clustering. If a sufficient number of data points are found within this "radius" from an arbitrary point, a new cluster is created, otherwise any data points within this "radius" become noise points (individual, unclustered markers).

Note that setting this property to a very low value may result in too many points classified as noise points instead of being aggregated in a cluster. Larger values can help resolve this problem, however, they may also produce bigger clusters than desired/expected.

Please note, that due to certain technical details, eps "radius" is approximate and is not pixel precise. In all classes inside `nokia.maps.clustering` package where we refer to eps or eps neighborhood of the given point, we assume a rectangular area with $2 * \text{eps}$ sides and with that given point as a center.

Default Value: 50

index: {[nokia.maps.clustering.Index](#)}

This property holds an indexing data structure to use along with the clustering data. It can lead to a significant improvement in the speed of marker data retrieval operations. If this property is not set, `ClusterProvider` uses an instance of [nokia.maps.clustering.Index](#) with the default [nokia.maps.clustering.RTreeConfiguration](#).

Default Value: `nokia.maps.clustering.Index`

max: {Number}

This property holds a value of the maximum zoom level at which to perform clustering.

Default Value: 20

min: {Number}

This property holds a value of the minimum zoom level at which to perform clustering.

Default Value: 0

minPts: {Number}

here

This property holds the smallest number of points that must be present within the eps distance of an arbitrary point to form a cluster. In other words every cluster should contain minimum minPts+1 points.

Default Value: 1

strategy: {Number}

This property specifies strategy to use during clustering. Possible values are:

- `nokia.maps.clustering.ClusterProvider.STRATEGY_DENSITY_BASED`
- `nokia.maps.clustering.ClusterProvider.STRATEGY_GRID_BASED`

Default Value: `nokia.maps.clustering.ClusterProvider.STRATEGY_DENSITY_BASED`

theme: {[nokia.maps.clustering.ITheme](#)}

This property holds an object that represents the display theme for clusters and noise points. By default, ClusterProvider uses an instance of [nokia.maps.clustering.MarkerTheme](#). It creates an SVG marker for each cluster. The marker displays a number reflecting the number of data points in the cluster. The size and color of the marker depends on that number as well.

Default Value: `nokia.maps.clustering.MarkerTheme`

Class: ColorizerTheme

This class is a member of [nokia.maps.clustering](#).

Class Summary

An instance of this class provides a colored representation for cluster and noise point markers on the map

[For full details, see [nokia.maps.clustering.ColorizerTheme](#)]

Table 72: Method Summary

Methods
getClusterPresentation (data) : { nokia.maps.map.Container } This method retrieves a container with markers corresponding to all the data points in a cluster (all the markers are set to have the same color).

here

Methods

`getNoisePresentation` (dataPoint) : {[nokia.maps.map.StandardMarker](#)}

This method retrieves a standard marker representing a noise point.

Class Description

Class provides a colored representation of cluster and noise points. This means that all markers in a cluster have the same color, but the color is different from cluster to cluster, all the data points share the same color, which is different from the cluster marker color. Note that noise points on the map are presented in the same way as standard markers, with light gray color. The class conforms to the interface [nokia.maps.clustering.ITheme](#).

Constructor Details

`nokia.maps.clustering.ColorizerTheme()`

This method creates an instance of `ColorizerTheme`.

Method Details

`getClusterPresentation`(data): {[nokia.maps.map.Container](#)}

This method retrieves a container with markers corresponding to all the data points in a cluster (all the markers are set to have the same color).

Parameters:

data: {[nokia.maps.clustering.IClusterPoint](#)[] | [nokia.maps.clustering.Cluster](#)}

A collection of objects representing data points or an instance of [nokia.maps.clustering.Cluster](#) class

Returns:

{[nokia.maps.map.Container](#)}

A container holding markers corresponding to all the data points in a cluster

`getNoisePresentation`(dataPoint): {[nokia.maps.map.StandardMarker](#)}

This method retrieves a standard marker representing a noise point.

here

Parameters:

dataPoint: {[nokia.maps.clustering.IClusterPoint](#)}

An object representing a data point (contains latitude, longitude and optionally weight)

Returns:

{[nokia.maps.map.StandardMarker](#)}

A standard marker positioned at the coordinates of the data point provided by the caller

Interface: IClusterPoint

This interface is a member of [nokia.maps.clustering](#).

Interface Summary

This interface defines properties for a data point that can be used in clustering.

[For full details, see [nokia.maps.clustering.IClusterPoint](#)]

Table 73: Property Summary

Properties
value: {Number} This property holds a numeric value that indicates the "weight" of the the data point within the cluster.

Directly Inherited Properties

Inherited from class [nokia.maps.geo.ICoordinate](#) :

[altitude](#), [altMode](#), [latitude](#), [longitude](#)

Interface Description

This interface defines properties for a data point that can be used in clustering. Each data point used for clustering purposes must conform to this interface.

Property Details

value: {Number}

here

This property holds a numeric value that indicates the "weight" of the the data point within the cluster. Weight is used to determine the position of the marker that represents the cluster in relation to the data points (markers) included in the cluster. Positioning the cluster marker by weight means that it is placed closer to the data points (markers) with the highest value (this property) than to any others. It is the default way to position the cluster marker, the other options being "center" and "first". "Center" ensures that the cluster marker is placed in the center of the cluster, while "first" means that the position of the cluster marker coincides with the first data point (marker) in the cluster.

You can determine the position of the cluster marker explicitly when initializing [nokia.maps.clustering.MarkerTheme](#). To do that, set the property "position" on an instance of `nokia.maps.clustering.MarkerTheme.Options#position` - the object is the argument to the `MarkerTheme` constructor. The example below demonstrates how to set weight-based positioning, using a constant defined on the class [nokia.maps.clustering.MarkerTheme](#):

```
var map = new nokia.maps.map.Display(...);
clusterProvider = new nokia.maps.clustering.ClusterProvider(map, {
  theme : new nokia.maps.clustering.MarkerTheme({
    position: nokia.maps.clustering.MarkerTheme.POSITION_WEIGHT_CENTER
  })
});
```

Note that if value is not set on any of the data points in the cluster, you can improve clustering performance by specifying "center" as the configuration setting for `MarkerTheme`, while achieving a similar visual effect as that created by "weight".

Default Value: 1

Interface: ITheme

This interface is a member of [nokia.maps.clustering](#).

Interface Summary

This interface defines how a cluster is to be shown on the map.

[For full details, see [nokia.maps.clustering.ITheme](#)]

Table 74: Method Summary

Methods
getClusterPresentation (dataPoints) : { nokia.maps.map.Object } This method retrieves an object that can be displayed on the map to represent a cluster.
getNoisePresentation (dataPoint) : { nokia.maps.map.Object }

here

Methods

This method retrieves an object that can be displayed on the map to represent a single noise point.

Interface Description

A variety of representations of marker cluster are possible. For example, a cluster can be shown as a rectangle with an outline boundary, or it can be displayed as a marker, etc. All cluster display themes must conform this interface to allow instances of `ClusteringProvider` to communicate with them.

Method Details

getClusterPresentation(dataPoints): {*nokia.maps.map.Object*}

This method retrieves an object that can be displayed on the map to represent a cluster.

Parameters:

dataPoints: {*nokia.maps.clustering.IClusterPoint*[] | *nokia.maps.clustering.Cluster*}

A collection of objects representing data points (each object contains latitude, longitude and optionally a `value` attribute to indicate the weight of the point); alternatively, the argument can be an instance of *nokia.maps.clustering.Cluster*

Returns:

{*nokia.maps.map.Object*}

An object that can be displayed on the map to represent a cluster

getNoisePresentation(dataPoint): {*nokia.maps.map.Object*}

This method retrieves an object that can be displayed on the map to represent a single noise point.

Parameters:

dataPoint: {*nokia.maps.clustering.IClusterPoint*}

An object representing a data point (contains latitude, longitude and optionally weight)

here

Returns:

`{nokia.maps.map.Object}`

An object that can be displayed on the map to represent a noise point

Class: Index

This class is a member of [nokia.maps.clustering](#).

Class Summary

This class holds clustering index based on R-Tree.

[For full details, see [nokia.maps.clustering.Index](#)]

Table 75: Method Summary

Methods
add (coordinate) This method adds a data point to the index.
clean () This method removes all the references to the data points from the index.
destroy () This method cleans up all internal objects and prepares the given instance of this class to be destroyed.
getNeighborhood (point, eps, seacrhAreaSize) This method finds all the data points which are in the rectangular area center of which is the given point, and both width and height are 2 * eps in size.
remove (coordinate) This method removes the data point specified by the caller from the index.

Class Description

Clustering index based on R-Tree.

Constructor Details

`nokia.maps.clustering.Index(config)`

This method creates new clustering index. It builds an [RTree](#) data structure on top of the clustering data, which can offer a significant improvement in the speed of data retrieval operations related to spatial objects.

here

Parameters:

config: `{nokia.maps.clustering.RTreeConfiguration}` [optional]

Configuration settings for RTree

Method Details

add(coordinate)

This method adds a data point to the index.

Parameters:

coordinate: `{nokia.maps.clustering.IClusterPoint}`

An object representing a data point (contains latitude and longitude)

clean()

This method removes all the references to the data points from the index. The effect is as if the index were created with an empty list of data points.

destroy()

This method cleans up all internal objects and prepares the given instance of this class to be destroyed.

getNeighborhood(point, eps, searchAreaSize)

This method finds all the data points which are in the rectangular area center of which is the given point, and both width and height are $2 * \text{eps}$ in size.

Parameters:

point: `{nokia.maps.clustering.IClusterPoint}`

A data point object whose neighborhood is to be established

eps: `{Number}`

Size of the neighborhood to find

here

`searchAreaSize:` `{Number}` [optional]

Size of the "world" in pixels; used in cases when there is a need to specify that the "world is round" and it is desirable to allow the search to continue on the opposite side of the map or the search area.

Returns:

`nokia.maps.util.RTreeRecord[]` An array of data points retrieved from the square area with 2 * `eps` sides and with the supplied point as a center; each point has the property `$coordinate` which holds a reference to the actual data point provided when calling the method [nokia.maps.clustering.Index#add](#)

remove(coordinate)

This method removes the data point specified by the caller from the index.

Parameters:

`coordinate:` `{nokia.maps.clustering.IClusterPoint}`

An object representing a data point (contains latitude and longitude)

Throws:

run-time error if no data point object is provided, or if it the supplied object does not conform to the interface [nokia.maps.clustering.IClusterPoint](#)

Class: MarkerTheme

This class is a member of [nokia.maps.clustering](#).

Class Summary

An instance of this class represents each cluster and noise point on the map as a marker.

[For full details, see [nokia.maps.clustering.MarkerTheme](#)]

here

Table 76: Property Summary

Properties
static <i>POSITION_CENTER</i>: {String} This constant provides a value for <code>nokia.maps.clustering.MarkerTheme.Options#position</code>.
static <i>POSITION_FIRST</i>: {String} This constant provides a value for <code>nokia.maps.clustering.MarkerTheme.Options#position</code>.
static <i>POSITION_WEIGHT_CENTER</i>: {String} This constant provides a value for <code>nokia.maps.clustering.MarkerTheme.Options#position</code>.

Table 77: Method Summary

Methods
<i>getClusterCoordinate</i> (data) : {<i>nokia.maps.geo.ICoordinate</i>} This method retrieves an object containing the geographic coordinates of the marker that represents a cluster.
<i>getClusterPresentation</i> (data) : {<i>nokia.maps.map.Marker</i>} This method retrieves a marker representing a cluster.
static <i>getColor</i> (clusterSize) : {String} This method retrieves a color definition as a hex string in the format #RRGGBB.
<i>getNoisePresentation</i> (dataPoint) : {<i>nokia.maps.map.StandardMarker</i>} This method retrieves a standard marker representing a noise point.

Class Description

This class conforms to the interface [*nokia.maps.clustering.ITheme*](#). Each cluster appears on the map as a circle showing a number indicating the size of the cluster. The color of the circle, its size and shadow depend on the size of the cluster. By contrast, noise points are presented as small blue dots, with white stroke color.

Constructor Details

`nokia.maps.clustering.MarkerTheme(position)`

This method creates an instance of `MarkerTheme`. It accepts configuration settings as an argument.

Parameters:

position: {String} [optional]

here

A value indicating the position of the cluster presenter marker; it should be one of the static variables `POSITION_CENTER`, `POSITION_FIRST` and `POSITION_WEIGHT_CENTER` defined in this class; if the method receives an unsupported position identifier, it uses `POSITION_WEIGHT_CENTER` by default

Property Details

`static POSITION_CENTER: {String}`

This constant provides a value for `nokia.maps.clustering.MarkerTheme.Options#position`. It ensures that the cluster marker is drawn in the center of the minimum rectangle containing all data points in the cluster.

`static POSITION_FIRST: {String}`

This constant provides a value for `nokia.maps.clustering.MarkerTheme.Options#position`. It ensures that the cluster marker is placed at the same location as the first data point in the cluster.

`static POSITION_WEIGHT_CENTER: {String}`

This constant provides a value for `nokia.maps.clustering.MarkerTheme.Options#position`. It ensures that the cluster marker is placed closer to the data points (markers) with the highest value ("weight") than to any others.

Method Details

`getClusterCoordinate(data): {nokia.maps.geo.ICoordinate}`

This method retrieves an object containing the geographic coordinates of the marker that represents a cluster.

Parameters:

`data:` `{nokia.maps.clustering.IClusterPoint[] | nokia.maps.clustering.Cluster}`

A collection of objects representing data points (contain latitude and longitude) or an instance of *nokia.maps.clustering.Cluster*; in addition, each point may contain a "value" attribute if you wish to position the

here

cluster marker according to the "weight" center of the cluster (see also [nokia.maps.clustering.IClusterPoint](#))

Returns:

{[nokia.maps.geo.ICoordinate](#)}

An object containing the geographic coordinates of the position of cluster marker

getClusterPresentation(data): {[nokia.maps.map.Marker](#)}

This method retrieves a marker representing a cluster.

Parameters:

data: {[nokia.maps.clustering.IClusterPoint](#)[] | [nokia.maps.clustering.Cluster](#)}

A collection of objects representing data points (contain latitude and longitude) or an instance of [nokia.maps.clustering.Cluster](#); in addition, each point may contain a "value" attribute if you wish to position the cluster marker according to the "weight" center of the cluster (see also [nokia.maps.clustering.IClusterPoint](#))

Returns:

{[nokia.maps.map.Marker](#)}

An instance of [nokia.maps.map.StandardMarker](#) showing a numeric value indicating the number of elements inside the cluster. The color of the marker also depends on the size of the cluster

static **getColor**(clusterSize): {String}

This method retrieves a color definition as a hex string in the format #RRGGBB. The retrieved color definition, depends on the cluster size.

- #76D100 - if cluster size is smaller than 10
- #FF6900 - if cluster size is bigger than 10, but less than 25
- #F03C00 - if cluster size is bigger than 25, but less than 50

here

- #B50015 - if cluster size is bigger than 100

Parameters:

clusterSize: {Number}
A value indicating the size of the cluster

Returns:

{String} An string containing the color definition in the format #RRGGBB

getNoisePresentation(dataPoint): {[nokia.maps.map.StandardMarker](#)}

This method retrieves a standard marker representing a noise point.

Parameters:

dataPoint: {[nokia.maps.clustering.IClusterPoint](#)}
An object representing a noise point, containing latitude, longitude and optionally weight

Returns:

{[nokia.maps.map.StandardMarker](#)}
A standard marker positioned at the location supplied by the argument

Class: Noise

This class is a member of [nokia.maps.clustering](#).

Class Summary

This class represents noise point on the map.

[For full details, see [nokia.maps.clustering.Noise](#)]

Table 78: Method Summary

Methods
getCoordinate () : { nokia.maps.geo.Coordinate } This method retrieves an object containing the geographic coordinates of the given noise point.

here

Methods

`getIcon` (theme, resetTheMarker)

This method retrieves the icon representing of the noise point on the map.

Class Description

This class represents a noise point - a point that is not part of a cluster and is indicated by an individual marker. Note that at the highest zoom level at which clustering applies (see also [nokia.maps.clustering.ClusterProvider](#)), it is possible that all or most data points are presented as noise points.

Constructor Details

`nokia.maps.clustering.Noise`(point)

This method creates an instance of Noise.

Parameters:

point: {[nokia.maps.clustering.IClusterPoint](#)} [optional]
 A data point object that provides the coordinates for a noise point

Method Details

`getCoordinate()`: {[nokia.maps.geo.Coordinate](#)}

This method retrieves an object containing the geographic coordinates of the given noise point.

Returns:

{[nokia.maps.geo.Coordinate](#)}
 An object containing the geographic coordinates of the noise point

`getIcon`(theme, resetTheMarker)

This method retrieves the icon representing of the noise point on the map.

Parameters:

theme: {[nokia.maps.clustering.ITheme](#)}
 A theme object to use when presenting cluster on the map

here

`resetTheMarker:` `{Boolean}` [optional]

A flag indicating whether the internal marker is to be updated (`true`) or whether the cached marker is to be retrieved (`false`; this optional argument can be used when the theme for existing clusters has been modified

Interface: RTreeConfiguration

This interface is a member of [nokia.maps.clustering](#).

Interface Summary

This interface defines configuration options for R-Tree.

[For full details, see [nokia.maps.clustering.RTreeConfiguration](#)]

Table 79: Property Summary

Properties
<code>algorithm</code>: <code>{Number}</code> This property specifies the partitioning algorithm for RTree.
<code>max</code>: <code>{Number}</code> This property holds the maximum number of objects per node.
<code>min</code>: <code>{Number}</code> This property holds the minimum number of objects per node.

Interface Description

This interface defines configuration options for [RTree](#). It is used as a template for configuration settings used when creating an instance of [nokia.maps.clustering.Index](#).

Property Details

`algorithm`: `{Number}`

This property specifies the partitioning algorithm for RTree.

The possible values are:

- 0 or `nokia.maps.util.RTree.QUADRATIC_PARTITIONING`
- 1 or `nokia.maps.util.RTree.LINEAR_PARTITIONING`
- 2 or `nokia.maps.util.RTree.LINEAR_MOD_PARTITIONING`

here

- 3 or `nokia.maps.util.RTree.DUMMY_PARTITIONING`

Default Value: 0

max: {Number}

This property holds the maximum number of objects per node. Its value must not be less than 2.

Default Value: 32

min: {Number}

This property holds the minimum number of objects per node. Its value must not be less than 1 or greater than $\text{max}/2$. The time needed for querying the index dependent on this property. A large min value reduces the height of the tree and improves space utilization. Too large a min, however, reduces the tree to a linear index, which has the same effect as having index at all. A small value of min on the other hand, causes a higher tree, partitioning the objects more. When choosing min too small, too many nodes have to be examined when searching the tree, increasing the required search time. A trade-off between a wide tree and a high tree has to be made. Therefore, the parameters min and max can be tuned as part of performance tuning.

Default Value: 12

Class: RectangleTheme

This class is a member of [*nokia.maps.clustering*](#).

Class Summary

This class creates a representation of each cluster on the map as a rectangle which contains all the data points that belong to the cluster.

Deprecated: Since 2.5.0

[For full details, see [*nokia.maps.clustering.RectangleTheme*](#)]

Table 80: Method Summary

Methods
<i>getClusterPresentation</i> (data) : { <i>nokia.maps.map.Rectangle</i> } This method retrieves the smallest rectangle that encloses all the data points in the cluster provided by the caller.

here

Methods

`getNoisePresentation` (dataPoint) : {*nokia.maps.map.StandardMarker*}

This method retrieves a standard marker representing a noise point.

Class Description

The class creates a representation of a cluster on the map, presenting all noise points as standard markers, while clusters are shown as instances of *nokia.maps.map.Rectangle*. The class conforms to the interface *nokia.maps.clustering.ITheme*.

Constructor Details

~~**`nokia.maps.clustering.RectangleTheme`**(-)~~

This method creates an instance of ~~*RectangleTheme*~~.

Deprecated: Since 2.5.0

Method Details

`getClusterPresentation`(data): {*nokia.maps.map.Rectangle*}

This method retrieves the smallest rectangle that encloses all the data points in the cluster provided by the caller.

Parameters:

data: {*nokia.maps.clustering.IClusterPoint*[] |
 nokia.maps.clustering.Cluster}

A collection of data point objects containing latitude and longitude, or an instance of *nokia.maps.clustering.Cluster* class

Returns:

{*nokia.maps.map.Rectangle*}

An object representing a rectangle that encloses all the data points in the cluster

`getNoisePresentation`(dataPoint): {*nokia.maps.map.StandardMarker*}

This method retrieves a standard marker representing a noise point.

here

Parameters:

dataPoint: {[nokia.maps.clustering.IClusterPoint](#)}

An object that represents the noise point for which to retrieve the marker (contains latitude, longitude and optionally weight)

Returns:

{[nokia.maps.map.StandardMarker](#)}

A standard marker positioned at the coordinates of the data point supplied by the caller

Namespace: dom

This namespace is a member of [nokia.maps](#).

Namespace Summary

This namespace implements support for managing HTML documents, including DOM-related events, and for HTML 5.

Namespace Description

This namespace implements support for managing HTML documents, including DOM-related events, and for HTML 5.

Class: DataTransfer

This class is a member of [nokia.maps.dom](#).

Class Summary

This class is a implementation of the HTML 5 DataTransfer interface.

[For full details, see [nokia.maps.dom.DataTransfer](#)]

Table 81: Property Summary

Properties
cursor: {String} This property defines the CSS cursor to be shown during non-lift dragging; the default value is "pointer".

here

Properties

dropEffect: {String}

This property names the currently selected operation type.

effectAllowed: {String}

This property names allowed operations.

files: {Object[]}

This property contains references to the files that are being dragged, if any.

lift: {Boolean}

This property can only be modified within a dragstart event, it indicates if a drag operation lifts the target or not.

readonly **ndt**: {Object}

This property holds a reference to a native data transfer object, if available.

realTime: {Boolean}

This property determines if all the drag-events are to be processed in real time or if thresholds are to be applied to prevent too many events and therefore an overall performance drop.

readonly **types**: {String[]}

This property holds an array of strings with the formats that were set in the dragstart event.

Table 82: Method Summary

Methods

addElement (element)

This method offers an alternative way of specifying how the user agent is to render the drag feedback.

allows (dropEffect) : {Boolean}

This method checks if the supplied drop effect is allowed or not.

clearData (format) : {*nokia.maps.dom.DataTransfer*}

This method removes the data of the specified format.

getData (format)

This method retrieves the data specified by the caller.

hasData (format)

This method checks if data matching the format specified by the caller exists.

setData (format, data)

This method adds the data specified by the caller.

setDragImage (image, left, top)

This method sets the element to be used to generate the drag feedback.

here

Class Description

This class is a browser-independent implementation of the HTML 5 interface [DataTransfer](#). The class extends the capabilities stipulated in the HTML 5 specification by allowing native JavaScript objects as values. If there is native support and the value is a string, then the data is copied into the native data transfer object.

Constructor Details

nokia.maps.dom.DataTransfer(nativeDataTransfer)

This method initializes a new instance of DataTransfer

Parameters:

nativeDataTrans- {Object} [optional]

fer:

The reference to a native data transfer object (requires browser support for native drag and drop)

Property Details

cursor: {String}

This property defines the CSS cursor to be shown during non-lift dragging; the default value is "pointer".

dropEffect: {String}

This property names the currently selected operation type. To succeed, the operation must be allowed by the attribute effectAllowed.

The value of the property can be set to change the selected operation. The possible values are none, copy, link, and move.

effectAllowed: {String}

This property names allowed operations.

The value of the property can be set to modify the list of allowed operations. The possible values are none, copy, copyLink, copyMove, link, linkMove, move, all, and uninitialized.

here

files: {Object[]}

This property contains references to the files that are being dragged, if any.

lift: {Boolean}

This property can only be modified within a `dragstart` event, it indicates if a drag operation lifts the target or not.

If an object is dragged, but not lifted, no `dragenter`, `dragover`, `dragleave` or `drop` events are dispatched. Dragging without lift occurs, for example, when the slider or the scale bar is moved, or when the map is panned. It means that something is dragged within a limited area and cannot leave that area -- for example, the thumb shift of a zoom slider cannot leave the zoom slider, therefore the zoom slider handles the movement of the thumb shift, and consequently updates the slider within the drag event, without notifying any other element in the Web site or outside the browser window.

In dragging without lift, the cursor is not defined by the `dropEffect`, but by the `cursor` property.

readonly **ndt:** {Object}

This property holds a reference to a native data transfer object, if available.

realTime: {Boolean}

This property determines if all the drag-events are to be processed in real time or if thresholds are to be applied to prevent too many events and therefore an overall performance drop. The property has the best effect if set directly within the `dragstart` event, otherwise it will only be partially "near-time".

Note that this property has no effect in native drag and drop and we recommended that you avoid real-time drag and drop operations if possible.

readonly **types:** {String[]}

This property holds an array of strings with the formats that were set in the `dragstart` event. In addition, if any files are being dragged, one of the types is the string "Files".

Method Details

addElement(element)

here

This method offers an alternative way of specifying how the user agent is to render the drag feedback. It adds an element to the `DataTransfer` object.

Parameters:

`element:` `{Node}`

The element to add as drag feedback for the user

`allows(dropEffect):` `{Boolean}`

This method checks if the supplied drop effect is allowed or not.

This is a proprietary member and not part of the implementation of W3C interface specifications.

Parameters:

`dropEffect:` `{String}`

The drop effect to check: copy, move or link

Returns:

`{Boolean}` A value is `true` if the effect is allowed, `false` otherwise

`clearData(format):` `{nokia.maps.dom.DataTransfer}`

This method removes the data of the specified format. It removes all data if the argument is omitted.

Parameters:

`format:` `{String}`

The format for which the data should be removed.

Returns:

`{nokia.maps.dom.DataTransfer}`

The data transfer object itself (`this`).

`getData(format)`

here

This method retrieves the data specified by the caller. If there is no matching data, an empty string is returned.

Parameters:

format: {String}
The format of the data to return

Returns:

An {Object} containing the data

hasData(format)

This method checks if data matching the format specified by the caller exists.

Parameters:

format: {String}
The format of the data to check for

Returns:

A {Boolean} value, `true` if the data in the given format is available; `false` otherwise.

setData(format, data)

This method adds the data specified by the caller.

Parameters:

format: {String}
The format of the data

data: {Object}
The data

Returns:

here

An instance of `{nokia.maps.dom.DataTransfer}` representing the data transfer object itself (`this`)

setDragImage(image, left, top)

This method sets the element to be used to generate the drag feedback. The element can be any element; if it is an `img`, then the user agent should use the element's image (at its intrinsic size) to generate the feedback, otherwise the user agent should base the feedback on the given element (but the exact mechanism for doing so is not specified).

Parameters:

image: `{node}`

The feedback element to be used as drag feedback

left: `{Object}` [optional, default: 0]

The horizontal pixel offset of the left border of the image relative to the cursor, a positive number moves the image to the right of the cursor, a negative number moves the image left to the cursor

top: `{Object}` [optional, default: 0]

The vertical pixel offset of the top border of the image relative to the cursor, a positive number moves the image to the bottom of the cursor, a negative number moves the image top to the cursor

Example:

```
// If "imageNode" is a DOM image with a size of 50 pixel,
// then this would attach this image to the cursor,
// align it 25 pixel to top-left
event.dataTransfer.setDragImage( imageNode, -25, -25 );
```

Interface: DragEvent

This interface is a member of [*nokia.maps.dom*](#).

Extends: [*nokia.maps.dom.Event*](#)

here

Interface Summary

This event is fired for all kind of drag operations.

[For full details, see [nokia.maps.dom.DragEvent](#)]

Table 83: Property Summary

Properties
dataTransfer: {Number} The property <code>dataTransfer</code> contains context information maintained during the entire dragging operation, from dragstart until the dragend event.

Directly Inherited Properties

Inherited from class [nokia.maps.dom.Event](#) :

[AT_TARGET](#), [bubbles](#), [BUBBLING_PHASE](#), [canBubble](#), [cancelable](#), [canSlicker](#), [CAPTURING_PHASE](#), [currentTarget](#), [defaultPrevented](#), [eventPhase](#), [namespaceURI](#), [nativeEvent](#), [page](#), [propagation](#), [PROPAGATION_OK](#), [PROPAGATION_STOP](#), [PROPAGATION_STOP_IMMEDIATE](#), [target](#), [timeStamp](#), [type](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.dom.Event](#) :

[cancel](#), [clone](#), [preventDefault](#), [preventUnload](#), [stopImmediatePropagation](#), [stopPropagation](#)

Interface Description

This interface represents events fired for all kind of drag operations. For more information about drag events please refer to the [nokia.maps.dom.DragEventTarget](#) class.

Note that this class represents an unimplemented interface. It is included here to document its properties. The class is derived from [nokia.maps.dom.Event](#), but, in addition to the properties of the parent class, it has the properties described here.

Property Details

`dataTransfer`: {Number}

The property `dataTransfer` contains context information maintained during the entire dragging operation, from dragstart until the dragend event. It can be used to store data while the drag

here

operation is ongoing. The object always has the same reference, therefore it is possible to modify its contents (see [nokia.maps.dom.DataTransfer](#)), but not to replace it with a new one.

Note that although according to the W3C specification, it is only possible to store binary safe strings in the data transfer object, this is not the case in this implementation. However, only strings may be passed from one window to another, and therefore adding a JavaScript object to the data transfer object prevents strings from being transferred into another window.

Interface: DragEventTarget

This interface is a member of [nokia.maps.dom](#).

Interface Summary

This class is a *virtual* interface described here for reference only.

[For full details, see [nokia.maps.dom.DragEventTarget](#)]

Table 84: Event Summary

Events
drag This event is fired at the target object of the dragstart event while a drag operation is in progress.
dragend This event is fired at the end of a drag operation at the object that was dragged (the target of the dragstart event).
dragenter This event is fired at an object if the mouse/finger is moved into the visible area of the object during a drag operation.
dragleave This event is fired at the current drop target if the the mouse/finger leaves the visible area of the target.
dragover This event is fired at the current drop target while the mouse/finger is on/above the drop target.
dragstart This event is fired at an object that has the property isDraggable set to true and after a mousedown/touch has occurred and the mouse/finger was moved at least three pixels.
drop This event is fired at the current drop target if the mouse button or finger is released on/above it, which means that the dragenter event and the dragover events were canceled, so their preventDefault methods were called and the allowedEffect property matches the dropEffect property.

here

Interface Description

Each class implementing this interface declares that it can act as the target for certain events. Each event in this interface (and therefore also within the classes implementing the interface) represents an event of a specific type.

The following example shows event handling for events of the type "click":

```
// Note that "obj" can be either a DOM node or any other JavaScript object.
var obj = nokia.maps.dom.EventTarget( {} );
obj.addListener("click", function (evt) {
  console.log("This is the '"+evt.type+"' event!");
});

obj.dispatch( new nokia.maps.dom.Event({
  type: "click"
}));
```

The example creates an instant of `nokia.maps.dom.EventTarget`, adds a listener for "click" events to it, and dispatches a "click" event to all registered listeners (in this case one) - the listeners receive the object that represents the target of the event.

For more information about dispatching events, please refer to the documentation of the [nokia.maps.dom.EventTarget](#).

Drag and drop in a nutshell

The following description discusses event handling for drag and drop events. These are likely to be the most complex events, although once you have understood them, you should find them quite straightforward to process. The W3C homepage offers a very detailed explanation of drag and drop - please see <http://dev.w3.org/html5/spec/dnd.html>. Our text attempts to capture the information you need in a nutshell, plus it covers some information about the non-standard *not lifted* dragging, which is an arbitrary extension to the W3C specification and should help you use drag and drop events for certain special cases (such as implementing sliders).

Drag and drop concerns seven events, three of them (`dragstart`, `drag` and `dragend`) are fired at the DOM node that is dragged and four (`dragenter`, `dragover`, `dragleave` and `drop`) are fired at the DOM node over which the source node is dragged or onto which the source node is dropped (this does not occur in *not lifted* dragging, but more about that later).

Before we start

First, however, please note that the following code is always expected to be executed before the small code examples we provide below:

```
// Create a few shortcuts.
var Page = nokia.maps.dom.Page,
    EventTarget = nokia.maps.dom.EventTarget;
```

here

```
// Query page support for the document. This is very
// important, but needs only to be done once.
Page(document);
```

Dragging Basics

The first step is to ensure that the DOM node to be dragged is draggable. The class [nokia.maps.dom.EventTarget](#) offers support in this respect through a method that takes care of all browser-specific details:

```
// Make a DOM node an event target and draggable.
EventTarget(node).enableDrag();
```

This simple code guarantees that the browser fires a `dragstart` event as soon as the user tries to drag node or any of its children. What is more, this is valid for touch screens (for example the iPhone). If the event is canceled, then dragging is disallowed, otherwise it is allowed (by default). While a drag operation is in progress, no other mouse, touch or keyboard events are fired.

Note that the above applies also to the `mouseup` and `click` events. You can get a `mousedown` event, but no `mouseup` or `click` event follow it if the `mousedown` event started a drag operation and `dragstart` event is not canceled with a call to `preventDefault()`. Therefore, it is best not only to register a listener for `mousedown` and `mouseup` (assuming you want to track these two events), but also for `dragend` in the *capture phase*. A `mousedown` followed by a `dragend` indicates the mouse button has been released - a `mouseup` is not generated. This means you can simply use the `dragend` as a `mouseup` event. If you received a `mousedown` at a node, you can be sure that you will receive a `dragend` at the same node at least within the *capture phase*.

Each of the seven drag events has a special property called `dataTransfer`, which holds an instance of [nokia.maps.dom.DataTransfer](#). This object contains information about the drag operation and is used to control it. In addition, the object can be used to transfer data between the different drag listeners, as it is guaranteed that the object itself is attached to all drag events from `dragstart` to `dragend`. However, the next drag sequence has a new `dataTransfer` object associated with it (so you cannot transfer data between to separate drag sequences).

Dragging from the point of view of drag target

The event `dragstart` has a property named `effectAllowed`. It determines what kind of operation can be performed on the dragged object, for example, whether the object can be copied, linked and/or moved. The value of this property is set by the event handler to one of the following strings: "uninitialized", "none", "copy", "copyLink", "copyMove", "all", "link", "linkMove" or "move". The default value is "uninitialized" and has the same meaning as "all".

The `dragstart` event allows you to define the visual feedback of the drag as well. There are two methods available on the data transfer object for this purpose: `setDragImage(node, offsetX, offsetY)` and `addElement(node)`. You can only have one drag image (a picture visible while

here

dragging is in progress), so a second call to `setDragImage()` overwrites the previously set image node. `addElement()` adds an unlimited number of additional elements to the visual feedback. The visual feedback (all added elements and/or the drag image) is rendered relative to the hot spot of the mouse cursor or the touch point of the finger on a touch screen and is shown stretching to the right and down from the mouse pointer or the dragging finger. You can fine-tune where the drag image appears via its offset coordinates. The offset has no effect for the elements added using the `addElement()`, but their style property `position` can be set to "absolute" and then they can be offset using the CSS properties `style.left` and `style.top`.

While dragging is in progress, the browser fires a "drag" event every few milliseconds at the source node that is being dragged. If the event is canceled (`preventDefault` is called), dragging is aborted, otherwise it continues. In most cases, the drag event can be ignored, except, for example, for *not lifted* dragging to which we return shortly.

When dragging is finished, the source node (the drag target) receives a `dragend` event with information reflecting the action that was performed when the dragged object reached the drop zone. The information is stored in the property `dropEffect` of the data transfer object and is "none", "copy", "move" or "link". If the drop was aborted or failed, the effect is none. This is important information in the example that follows, because if the dragged element was not moved, the node must be re-inserted into the document at the position it occupied before dragging began:

```
// This function makes an image node draggable.
var makeImageDraggable = function ( node ) {
// Create a closure so that every call to this method
// can handle it's own node.
(function (node) {
    // Make the image draggable.
    EventTarget(node).enableDrag();

    // These variables are used later to put the image
    // back where it was should dragging fail.
    var sourceNodeParent;
    var sourceNodeNextSibling;

    // Register a dragstart listener.
    node.addListener("dragstart", function (evt) {
// Remember the parent node and next sibling of the image.
sourceNodeParent = node.parentNode;
sourceNodeNextSibling = sourceNode.nextSibling;

// Detach the image from the document.
sourceNodeParent.removeChild (node);

// Attach the image to the cursor so that it is dragged around.
evt.dataTransfer.setDragImage( sourceNode, -evt.targetX || 0, -evt.targetY
|| 0);

// Allow all kinds of drag operations.
```

```
evt.dataTransfer.effectAllowed = "all";

// Prevent any other drag handlers from receiving this event,
// but do not cancel the event, because that would abort
// dragging.
evt.stopImmediatePropagation();
}, false);

// Register a dragend listener.
node.addListener("dragend", function (evt) {
// If the image was not moved somewhere else
if (evt.dataTransfer.dropEffect !== "move") {
// we have to move the image back to where it was.
if (sourceNodeNextSilbling)
sourceNodeParent.insertBefore(node, sourceNodeNextSilbling);
else
sourceNodeParent.appendChild(node);
}
// Note: If the image was moved, we don't need to do anything!

// Let's cancel the event now that we have processed it. The event
// must not be processed by the browser or any other listener.
event.cancel();
}, false);
})(node);
};
```

So far we have explained one aspect of drag and drop: what you need to do if you want to make a DOM node draggable. Next, we consider how to create a drop zone, where the dragged object can be dropped. Our next example extends the previous one by allowing the image to be dropped anywhere within the document.

The drag from the drop target perspective

A dropzone can be any DOM node or JavaScript object that is an instance of [nokia.maps.dom.EventTarget](#). Whenever the mouse is moved onto such a node, the browser fires a `dragenter` event at that node. If the event is not canceled (by calling `preventDefault`), the browser estimates that a drop is not handled by the node and disallows a drop to it. If the event is canceled (`preventDefault` has been called), the listener must set the property `dropEffect` on the data transfer object to indicate the operation that the node wants to perform with the dragged element. If an image is dragged, `dropEffect` must be said to "move".

The next event to be fired is `dragover`. It is fired regularly like the `drag` event, but it is fired at the current drop target. This event must be canceled (by calling `preventDefault`) and the property `dropEffect` on the data transfer object must be updated to indicate the desired operation ("none", "copy", "link" or "move"). If the event is not canceled, the drop is denied and the `dropEffect` is set to "none".

Finally if the user releases the mouse button or finger (touch screen) above a drop target that has set the `dropEffect` to "copy", "link" or "move", a drop event is fired at the drop target. The drop

here

handler executes the required drop effect and cancels the event (by calling `preventDefault`). If the event is not canceled, the browser automatically executes the drop action (which only works for very specific nodes).

The following code shows a complete drag and drop example for the image:

```
<html>
<head>
  <!-- please load the API here -->
  <script language="JavaScript">
    // NOTE: This is very important, otherwise no event handling at all
    // is possible!
    var page = nokia.maps.dom.Page(document);

    // Create a shortcut to the event target class.
    var EventTarget = nokia.maps.dom.EventTarget;

    // This function makes an image node draggable.
    var makeImageDraggable = function ( node ) {
      // Create a closure so that every call to this method
      // can handle its own node.
      (function (node) {
        // Make the image draggable.
        EventTarget(node).enableDrag();

        // These variables are used later to put the image
        // back to where it was originally, if dragging fails.
        var sourceNodeParent;
        var sourceNodeNextSibling;

        // Register a dragstart listener.
        node.addListener("dragstart", function (evt) {
          // Remember the parent node and next sibling of the image.
          sourceNodeParent = node.parentNode;
          sourceNodeNextSilbing = node.nextSilbing;

          // Detach the image from the document
          sourceNodeParent.removeChild (node);

          // Attach the image to the cursor so that it is dragged around.
          evt.dataTransfer.setDragImage( node, -evt.targetX || 0, -evt.targetY ||
0);

          // Allow all drag operations.
          evt.dataTransfer.effectAllowed = "all";

          // Add the image node into the data transfer for the drop zone.
          evt.dataTransfer.setData("image/gif", node);

          // Prevent that any other drag handler to receive this event,
          // but we must not cancel the event, otherwise the
          // dragging is aborted.
          evt.stopImmediatePropagation();
        }, false);
```

here

```
// Register a dragend listener.
node.addListener("dragend", function (evt) {
    // If the image was not moved somewhere else
    if (evt.dataTransfer.dropEffect!="move") {
        // we have to move the image back to where it was.
        if (sourceNodeNextSilbing)
            sourceNodeParent.insertBefore(node, sourceNodeNextSilbing);
        else
            sourceNodeParent.appendChild(node);
    }

    // Note: If the image was moved, we don't need to do anything!

    // Let's cancel the event now that we have processed it. The event
    // must not be processed by the browser or any other listener.
    evt.cancel();
}, false);
})(node);
};

function initDnD() {
    // First of all make the image draggable.
    makeImageDraggable (document.getElementById("draggableImage") );

    // Get the DOM node for our drop zone.
    var dropZone = EventTarget(document.getElementById("dropzone"));

    // If the draggable image is dragged into our drop zone
    dropZone.addListener("dragenter", function (evt) {
        // Allow all GIF images to be moved here.
        if (evt.dataTransfer.hasData("image/gif")) {
            evt.dataTransfer.dropEffect = "move";
            evt.cancel();
        }
    }, false);

    // While the draggable image is moved onto the drop zone
    dropZone.addListener("dragover", function (evt) {
        // Allow all GIF images to be moved here.
        if (evt.dataTransfer.hasData("image/gif")) {
            evt.dataTransfer.dropEffect = "move";
            evt.cancel();
        }
    }, false);

    // If the draggable image is dropped onto our drop zone
    dropZone.addListener("drop", function (evt) {
        // If the GIF image is dropped here, add it into our drop zone
        // and set the drop effect to "move".
        if (evt.dataTransfer.hasData("image/gif")) {
            dropZone.appendChild (evt.dataTransfer.getData("image/gif"));
            evt.dataTransfer.dropEffect = "move";
            evt.cancel();
        }
    }, false);
}
```

here


```

}
</script>
</head>
<body onload="initDnD()">
  <div>
    
  </div>
  <div id="dropzone" style="width:400px; height:300px; background-
color:#eee;">
    <p>Drop the image here</p>
  </div>
</body>
</html>

```

The *not lifted* drag and drop

The *not lifted* drag and drop is a simplified and optimized drag and drop. It is useful if you want to make draggable an element that is confined to a specific area and cannot leave that area, such as the thumb shift of a scroll bar -- once dragging begins, it must not follow the mouse freely, but must move only along the scroll bar. You can make this happen by setting the property `lift` on the data transfer object of the `dragstart` event to `false`. This causes three things:

1. The `dragenter`, `dragover`, `dragleave` and `drop` events are no longer be fired, therefore there is no drop zone for the dragged element.
2. The cursor visible while dragging can be defined by the listener of the `dragstart` and `drag` event, ignoring the `effectAllowed` and `dropEffect` properties. This can be done by modifying the property `cursor` property of the `dataTransfer` object.
3. A *not lifted* drag is always a pure *soft event*, which means simulated. The native browser drag and drop events are not be used, therefore such an event is confined to the document in which it is fired and no other element or external application can 'notice' the drag.

The following example shows how *not lifted* drag can be used to implement a slider:

```

<html>
<head>
  <!-- please load API here -->
  <script language="JavaScript">
    // NOTE: This is very important, otherwise no event handling is possible!
    var page = nokia.maps.dom.Page(document);

    // Create a shortcut to the event target class.
    var EventTarget = nokia.maps.dom.EventTarget;

    function initDnD() {
      // Let's make the scrollbar and the thumb shift draggable.
      var scrollbar = EventTarget(page.$("scrollbar") ).enableDrag();
      var knob = EventTarget(page.$("knob") ).enableDrag();
    }
  </script>
</head>
<body>
  <div>
    
  </div>
  <div id="dropzone" style="width:400px; height:300px; background-
color:#eee;">
    <p>Drop the image here</p>
  </div>
</body>
</html>

```

here

```
// Attach a dragstart listener to the scrollbar.
scrollbar.addListener("dragstart", function (evt) {
// If the knob is being dragged.
if (evt.target===knob) {
// Remember that the thumb shift is dragged and set the value to the
offset
// of the dragstart relative to the top-left corner of the target.
evt.dataTransfer.setData("application/slider", evt.targetY || 0);

// Switch to not lifted dragging and make the cursor a hand.
evt.dataTransfer.lift = false;
evt.dataTransfer.cursor = "hand";
} else
// If the user tries to drag the scalebar, cancel the event and
// disallow dragging
evt.cancel();
}, false);

// Attach a drag listener to the scrollbar.
scrollbar.addListener("drag", function (evt) {
// If the knob is dragged
if (evt.dataTransfer.hasData("application/slider")) {
var targetY = evt.dataTransfer.getData("application/slider");

// Calculate the absolute position of the scrollbar
// within the document.
var scrollbarPos = page.getClientRect(scrollbar);

// Calculate the position of the thumb shift relative
// to the scrollbar, parallel to the mouse cursor.
var y = evt.pageY - scrollbarPos.top - targetY;

// Do not allow the thumb shift to be moved outside of the scrollbar.
if (y<0) y = 0;
if (y+knob.offsetHeight >= scrollbar.offsetHeight)
y = scrollbar.offsetHeight - knob.offsetHeight - 1;

// Move the thumb shift.
knob.style.top = y+"px";
}
}, false);
}
</script>
</head>
<body onload="initDnD()">
<div id="scrollbar" style="position:absolute; top:10px; left:10px;
width:30px; height: 200px; background-color:#eee; border:1px solid black;">
<div id="knob" style="position:absolute; top:0; left:3px; width:24px;
height:40px; background-color:black;"> </div>
</div>
</body>
</html>
```

Event Details

drag

This event is fired at the target object of the `dragstart` event while a drag operation is in progress. This event is fired every few milliseconds during the drag operation. The `relatedTarget` property contains a reference to the object that is currently under the mouse/finger (only for simulated drag events, not for the native ones).

Please refer to the class documentation of the [nokia.maps.dom.DragEvent](#) for a more detailed description of this and other drag events.

Event Handler Parameters:

evt	{ nokia.maps.dom.DragEvent }
	An object representing the event

dragend

This event is fired at the end of a drag operation at the object that was dragged (the target of the `dragstart` event). The event is fired if the drop was aborted, failed or successful. The event signals the end of a drag operation. The state of the drop operation can be read from the `dropEffect` which can be "none", "move", "copy" or "link" and reflects the action performed at the drop target. This information can be used either to remove the dragged object (for "move") or to do nothing (for the other drop effects).

Please refer to the class documentation of the [nokia.maps.dom.DragEvent](#) for a more detailed description of this and other drag events.

Event Handler Parameters:

evt	{ nokia.maps.dom.DragEvent }
	An object representing the event

dragenter

This event is fired at an object if the mouse/finger is moved into the visible area of the object during a drag operation. If the event is **canceled** (`preventDefault` method was called at the event object),

here

a drop at the target is allowed, otherwise a drop is cannot occur. If the drop is allowed, the target becomes the current drop target.

Note that despite the event name, its behavior is more similar to that of `mouseover` than `mouseenter`.

Please refer to the class documentation of the [nokia.maps.dom.DragEvent](#) for a more detailed description of this and other drag events.

Event Handler Parameters:

evt	{ nokia.maps.dom.DragEvent }
	An object representing the event

dragleave

This event is fired at the current drop target if the the mouse/finger leaves the visible area of the target. Note that despite the name of the event, its behavior is more like that one of `mouseout` than `mouseleave`.

Please refer to the class documentation of the [nokia.maps.dom.DragEvent](#) for a more detailed description of this and other drag events.

Event Handler Parameters:

evt	{ nokia.maps.dom.DragEvent }
	An object representing the event

dragover

This event is fired at the current drop target while the mouse/finger is on/above the drop target. Note that the `dragenter` event must be canceled (`preventDefault` must be called) to make the object the current drop target. So if the `dragenter` event is not canceled, then no `dragover` events are fired at all. This event is fired every few milliseconds as long as the mouse cursor is on/above the drop target. If this event is not canceled (via `preventDefault`) the `dropEffect` property is set to "none", which disallows a drop into the drop target. If the event is canceled and the `dropEffect` is not "none", then a drop into this drop target is allowed.

here

Note that in spite of the name of this event, its behavior is very different from the behavior of the `mouseover` event.

Please refer to the class documentation of the [nokia.maps.dom.DragEvent](#) for a more detailed description of this and other drag events.

Event Handler Parameters:

evt {[nokia.maps.dom.DragEvent](#)}

An object representing the event

dragstart

This event is fired at an object that has the property `isDraggable` set to `true` and after a `mousedown/touch` has occurred and the mouse/finger was moved at least three pixels. If the event is not canceled (`preventDefault` of the event was not called), the drag operation is permitted. The property `dataTransfer` can be used to keep track of information while a drag operation is in progress.

Please refer to the class documentation of the [nokia.maps.dom.DragEvent](#) for a more detailed description of this and other drag events.

Event Handler Parameters:

evt {[nokia.maps.dom.DragEvent](#)}

An object representing the event

drop

This event is fired at the current drop target if the mouse button or finger is released on/above it, which means that the `dragenter` event and the `dragover` events were canceled, so their `preventDefault` methods were called and the `allowedEffect` property matches the `dropEffect` property.

If the event is not canceled (`preventDefault` is called) and the drop target is a text field (for example, a `textarea` or `input` element), then the content of the "text/plain" format is inserted into the area to which the mouse point is pointing or added to the end of that area. Otherwise, the `dropEffect` is set to "none" and no action takes place. If `preventDefault` was called, the sequence described above does not take place.

here

Please refer to the class documentation of the [nokia.maps.dom.DragEvent](#) for a more detailed description of this and other drag events.

Event Handler Parameters:

evt {[nokia.maps.dom.DragEvent](#)}

An object representing the event

Class: Event

This class is a member of [nokia.maps.dom](#).

Class Summary

This class implements the W3C Event interface.

[For full details, see [nokia.maps.dom.Event](#)]

Table 85: Property Summary

Properties
readonly AT_TARGET : {Number} This field is an event phase identifier, indicating the target phase, which means it is being evaluated at the event target.
bubbles : {Boolean} This property indicates whether the event is a bubbling event (true) or not (false).
readonly BUBBLING_PHASE : {Number} This field is an event phase identifier, indicating the bubbling phase.
canBubble : {Boolean} This property indicates whether the event can bubble (true) or not (false).
cancelable : {Boolean} This property indicates whether or not an event can have its default action canceled (prevented).
canSicker : {Boolean} This property indicates whether the event can sick (whether in the capture phase, listeners can be iterated from the top down) or not.
readonly CAPTURING_PHASE : {Number} This field is an event phase identifier, indicating the capture phase.
currentTarget : { nokia.maps.dom.EventTarget } This property indicates the object whose eventListeners are currently being processed.

here

Properties

readonly *defaultPrevented*: {Boolean}

This property indicates whether `preventDefault()` has been called for this event.

eventPhase: {Number}

This property indicates the phase in the event flow the given event has reached.

namespaceURI: {String}

This property holds the namespace URI associated with the event or `null` if it is unspecified.

nativeEvent: {Event}

This property holds a reference to the native DOM event on which the given normalized event object is based.

page: {*nokia.maps.dom.Page*}

This property holds a reference to the page to which the given event relates.

propagation: {Number}

This property indicates the status of event propagation.

PROPAGATION_OK: {Number}

This field is an identifier, indicating that the propagation of the event has not been stopped.

PROPAGATION_STOP: {Number}

This field is an identifier, indicating that the propagation of the event has been stopped.

PROPAGATION_STOP_IMMEDIATE: {Number}

This field is an identifier, indicating that the propagation of the event has been stopped with immediate effect.

target: {*nokia.maps.dom.EventTarget*}

This property holds the target object of the event.

timeStamp: {Number}

This property specifies the time at which the event was created in milliseconds relative to 1970-01-01T00:00:00Z.

type: {String}

This property holds the local name of the event type.

Table 86: Method Summary

Methods

cancel () : {*nokia.maps.dom.Event*}

This method cancels the given event.

clone () : {*nokia.maps.dom.Event*}

This method clones the given event without cloning the properties (non-recursive cloning only).

preventDefault () : {*nokia.maps.dom.Event*}

here

Methods

This method cancels the default action for the given event.

`preventUnload` (msg) : {*nokia.maps.dom.Event*}

This method can be used only when the given event is of the type "beforeunload" to ask the user to confirm whether he wants to leave the page.

`stopImmediatePropagation` () : {*nokia.maps.dom.Event*}

This method prevents other event listeners from being triggered and, in contrast to `Event.stopPropagation()`, its effect is immediate.

`stopPropagation` () : {*nokia.maps.dom.Event*}

This method prevents other event listeners from being triggered, but its effect is deferred until all event listeners attached on the `Event.currentTarget` have been triggered.

Class Description

This class implements the W3C interface *Event*.

For more details about event dispatch please refer to the documentation of the class *nokia.maps.dom.EventTarget*.

Constructor Details

`nokia.maps.dom.Event`(defaults)

Creates a *Event* instance.

Parameters:

defaults: {Object} [optional]

If this argument is provided, all properties from it are copied into the given event object.

Property Details

readonly **`AT_TARGET`**: {Number}

This field is an event phase identifier, indicating the target phase, which means it is being evaluated at the event target.

`bubbles`: {Boolean}

This property indicates whether the event is a bubbling event (`true`) or not (`false`).

here

readonly **BUBBLING_PHASE**: {Number}

This field is an event phase identifier, indicating the bubbling phase.

canBubble: {Boolean}

This property indicates whether the event can bubble (`true`) or not (`false`).

This is a proprietary member and not part of the W3C interface specification this class implements.

cancelable: {Boolean}

This property indicates whether or not an event can have its default action canceled (prevented). If the default action can be canceled, the value is `true`, otherwise it is `false`.

canSicker: {Boolean}

This property indicates whether the event can sicker (whether in the capture phase, listeners can be iterated from the top down) or not.

This is a proprietary member and not part of the W3C interface specification this class implements.

readonly **CAPTURING_PHASE**: {Number}

This field is an event phase identifier, indicating the capture phase.

currentTarget: {[nokia.maps.dom.EventTarget](#)}

This property indicates the object whose eventListeners are currently being processed. This is particularly useful during the capture and bubbling phases to identify where the event is in the [propagation path](#).

readonly **defaultPrevented**: {Boolean}

This property indicates whether `preventDefault()` has been called for this event.

eventPhase: {Number}

here

This property indicates the phase in the event flow the given event has reached. The value is one of: CAPTURING_PHASE, AT_TARGET, or BUBBLING_PHASE.

namespaceURI: {String}

This property holds the namespace URI associated with the event or `null` if it is unspecified.

nativeEvent: {Event}

This property holds a reference to the native DOM event on which the given normalized event object is based. The property is not W3C-conform and is not part of the W3C specification this class implements. It contains either `null` or a reference to the native event as sent by the browser.

page: {*nokia.maps.dom.Page*}

This property holds a reference to the page to which the given event relates.

Note that this property is not part of the W3C interface specifications this class implements.

propagation: {Number}

This property indicates the status of event propagation. Its possible values are PROPAGATION_OK, PROPAGATION_STOP or PROPAGATION_STOP_IMMEDIATE.

This is a proprietary member and not part of the W3C interface specification this class implements.

PROPAGATION_OK: {Number}

This field is an identifier, indicating that the propagation of the event has not been stopped.

This is a proprietary member and not part of the W3C interface specification this class implements.

PROPAGATION_STOP: {Number}

This field is an identifier, indicating that the propagation of the event has been stopped.

This is a proprietary member and not part of the W3C interface specification this class implements.

PROPAGATION_STOP_IMMEDIATE: {Number}

here

This field is an identifier, indicating that the propagation of the event has been stopped with immediate effect.

This is a proprietary member and not part of the W3C interface specification this class implements.

target: {*nokia.maps.dom.EventTarget*}

This property holds the target object of the event.

timeStamp: {Number}

This property specifies the time at which the event was created in milliseconds relative to 1970-01-01T00:00:00Z.

type: {String}

This property holds the local name of the event type.

Method Details

cancel(): {*nokia.maps.dom.Event*}

This method cancels the given event. It stops the event propagation immediately and prevents the default action.

This is a proprietary member and not part of the W3C interface specification this class implements.

Returns:

{*nokia.maps.dom.Event*}

A reference to this event

clone(): {*nokia.maps.dom.Event*}

This method clones the given event without cloning the properties (non-recursive cloning only).

This is a proprietary member and not part of the W3C interface specification this class implements.

Returns:

{*nokia.maps.dom.Event*}

A reference to the cloned event

here

preventDefault(): {*nokia.maps.dom.Event*}

This method cancels the default action for the given event. Calling this method for a non-cancelable event has no effect.

Returns:

{*nokia.maps.dom.Event*}

A reference to this event

preventUnload(msg): {*nokia.maps.dom.Event*}

This method can be used only when the given event is of the type "beforeunload" to ask the user to confirm whether he wants to leave the page.

Note that this method can be useful in debugging to ensure that everything is cleared up when the user has left the page. However, bear in mind that the method cannot be used to prevent the user from leaving the page.

This is a proprietary member and not part of the W3C interface specification this class implements.

Parameters:

msg: {String}

The message the browser should display in the confirmation dialog

Returns:

{*nokia.maps.dom.Event*}

A reference to the current event

stopImmediatePropagation(): {*nokia.maps.dom.Event*}

This method prevents other event listeners from being triggered and, in contrast to `Event.stopPropagation()`, its effect is immediate. Once it has been called, further calls to this method have no effect.

Returns:

{*nokia.maps.dom.Event*}

here

A reference to this event

stopPropagation(): {[nokia.maps.dom.Event](#)}

This method prevents other event listeners from being triggered, but its effect is deferred until all event listeners attached on the `Event.currentTarget` have been triggered. Once it has been called, further calls to this method have no effect.

Returns:

{[nokia.maps.dom.Event](#)}

A reference to this event

Class: EventTarget

This class is a member of [nokia.maps.dom](#).

Class Summary

This class implements the W3C DOM Level 3 interface [EventTarget](#) and extends it with a number of features.

[For full details, see [nokia.maps.dom.EventTarget](#)]

Table 87: Property Summary

Properties
readonly isDraggable : {Boolean} This property indicates if the given event target is draggable and may receive a dragstart, drag and dragend events (true); false otherwise.
readonly isEventTarget : {Boolean} This property indicates whether an object implements the nokia.maps.dom.EventTarget interface (true) or not (false).
parentNode : { nokia.maps.dom.EventTarget } This property holds a reference to the parent node of the event target (if set).
parentNodes : {Object} This property is designed to hold a hash table that contains the "namespaceURI" as key and the corresponding value is the parent node of the given event target.

here

Table 88: Method Summary

Methods
<i>addListener</i> (type, callback, useCapture) : {Object} This method registers an event listener.
<i>addListenerNS</i> (namespaceURI, type, callback, useCapture) : {<i>nokia.maps.dom.EventTarget</i>} This method registers an event listener for events originating from a specific namespace.
<i>addListeners</i> (obj) This method provides a means of registering multiple event listeners.
static <i>catchException</i> (enable) This method enables exception catching in the event dispatcher to make it possible to ignore errors in event handlers or it disables exception catching in the event dispatcher for debugging purposes.
<i>disableDrag</i> () : {<i>nokia.maps.dom.EventTarget</i>} This method disables dragging of this event target, causing this event target not to receive dragstart, drag and dragend events.
<i>disableUserSelect</i> () : {<i>nokia.maps.dom.EventTarget</i>} This method prevents user selection of text and elements within the event target and prevents the magnifier on the iPhone or other similar mobile devices.
<i>dispatch</i> (evt) : {Boolean} This method dispatches an event.
static <i>dispatchEvent</i> (target, event, dispatchPath) : {Boolean} This method dispatches an event to the given event target.
<i>enableDrag</i> () : {<i>nokia.maps.dom.EventTarget</i>} This method enables dragging of this event target and disables the user selection, allowing this event target to receive dragstart, drag and dragend events.
<i>enableUserSelect</i> () : {<i>nokia.maps.dom.EventTarget</i>} This method allows the user to select text and elements within the event target and allows the magnifier on the iPhone or other similar mobile devices.
<i>hitTest</i> (pageX, pageY) : {Boolean} This method tests if the given x/y position relative to the document lies within the outer bounding box of the node corresponding to the given EventTarget object.
<i>insertListener</i> (type, callback, useCapture) : {Object} This method registers an event listener as the first listener in the listener chain.
<i>insertListenerNS</i> (namespaceURI, type, callback, useCapture) : {<i>nokia.maps.dom.EventTarget</i>} This method registers an event listener for events originating from a specific namespace as the first in the listener chain.
<i>removeAllListeners</i> () This method remove all registered listeners from current EventTarget

here

Methods

`removeListener` (type, listener, useCapture) : {*nokia.maps.dom.EventTarget*}

This method removes an event listener.

`removeListenerNS` (namespaceURI, type, listener, useCapture) : {*nokia.maps.dom.EventTarget*}

This method removes an event listener for events originating from a specific namespace.

Class Description

This class implements the W3C DOM Level 3 interface *EventTarget* and extends it with a number of features. It can be used as a mixin for any JavaScript class or it can be applied at runtime to any DOM node or JavaScript object by simply casting the node/object to *EventTarget*. Note that for the sake of compatibility the W3C, some methods have been renamed. For example, `addEventListener()` becomes `addListener()`.

Usage example:

```
// Shortcut for easier code.
var EventTarget = nokia.maps.dom.EventTarget;

// Cast a DOM node into an valid nokia.maps.dom.EventTarget
// Note: The DOM node is modified by the EventTarget interface,
// therefore a second cast is not necessary.
var node = document.getElementById("whatever");
EventTarget(node);

// You can also cast normal JavaScript objects including bubbling
// and capture phases and all that DOM supports:
var myObject = {},
    myParentObject = {};
EventTarget(myObject).parentNode = EventTarget(myParentObject);

// Now you can fire an event at "myObject" that will
// bubble to "myParentObject":
myParentObject.addListener("customEvent", function (evt) {
    alert( evt.type );
});
myObject.dispatch(
    new nokia.maps.dom.Event({
        type: "customEvent"
    })
);

// Note: It is also possible to set the parentNode property of
// a JavaScript object to a DOM node. Events then flow
// into the DOM tree, although not at native level.
```

here

EventTarget offers access to the nokia.maps DOM helpers and normalizes browser event handling for this target. By default an event target is not draggable, but selectable. If this interface is implemented as a mixin, the property `isEventTarget` must be added and set to `true`.

MooTools compatibility notes

If you have problems with MooTools then a possible fix is to include MooTools after this API is fully loaded and to replace every occurrence of "addListener" in the MooTools library with, for example, "mooAddListener", and every occurrence of "removeListener" with, for example, "mooRemoveListener". You need to do this also in all libraries that are based on MooTools. The reason is that MooTools modifies the prototypes of some native JavaScript and DOM objects and therefore causes a collision with extensions used by this API.

Note that the name clash arises, because MooTools modifies not only those DOM nodes that it uses, but patches all instances of all objects, including native browser objects. To avoid the name clashes, you must rename the MooTools methods "addListener" and "removeListener", and load MooTools after this API so that this API can keep references to the original native browser methods instead of referring to the methods modified by MooTools.

The listener queue

The listeners to events are called in reverse registration order, which means that the listener registered last is the first to be notified of an event. Therefore, if you want a listener to be notified last in an existing queue, you must register it as the first listener in the chain. This can be done by using the arbitrary non-standard method `insertListener()` or `insertListenerNS()` defined on this interface (EventTarget).

Note that applying this interface to a DOM node, causes methods and properties to be copied to the DOM node that might cause collisions with other frameworks. Therefore before combining [nokia.maps.dom.EventTarget](#) with any other framework, check carefully that name conflicts do not arise.

Event flow in a nutshell

Events normally trickle and bubble through the hierarchy in a DOM tree, where each node can only have one parent node (referenced by the property `parentNode`). For example, if the user clicks on a DOM node, then the browser builds a [propagation path](#) by going from the node that was the target of the click via the `parentNode` up to the root node (the HTML tag), which does not have a parent node assigned to propagation path. The browser uses the propagation path to dispatch the event to each of the nodes in it. First, in the *capture phase*, the chain is iterated from the root node (the HTML node) down to the target node (which the user has hit with it's click). Subsequently, in the *bubbling phase*, the event is dispatched to each node again, this time starting from the target node, working back up to the root node. Thus, each node receives the event twice, once within the *capture phase* and once within the *bubbling phase*. Normally events are processed within the *bubbling phase*, but

here

there might be reasons to process them in the *capture phase*, for example to stop the events from reaching other listeners, because all events can be stopped by any listener at any node within the propagation path.

The registered event listeners must be limited to specific event types either in the *capture phase* or the *bubbling phase*. Additionally, they can be registered for all events of a certain type or limited to events of a certain type that originate from a specific namespace (see [nokia.maps.dom.EventTarget#addListenerNS](#)). In the latter case, the listener is notified only if the namespaceURI of the event is set to the desired namespace. This can be used to separate arbitrary application events from other events, such as standard W3C events.

For a more comprehensive description of how events are dispatched, please see the W3C documentation [DOM level 3 event flow](#).

Constructor Details

nokia.maps.dom.EventTarget(obj)

The class constructor mixes the functionality of EventTarget into the argument object.

Parameters:

obj: {Object}
The object to be extended with EventTarget functionality

Property Details

readonly **isDraggable**: {Boolean}

This property indicates if the given event target is draggable and may receive a dragstart, drag and dragend events (true); false otherwise.

See: [nokia.maps.dom.EventTarget#enableDrag](#)
[nokia.maps.dom.EventTarget#disableDrag](#)

readonly **isEventTarget**: {Boolean}

This property indicates whether an object implements the [nokia.maps.dom.EventTarget](#) interface (true) or not (false).

parentNode: {[nokia.maps.dom.EventTarget](#)}

here

This property holds a reference to the parent node of the event target (if set).

parentNodes: {Object}

This property is designed to hold a hash table that contains the "namespaceURI" as key and the corresponding value is the parent node of the given event target. The use of the property is optional.

Method Details

addListener(type, callback, useCapture): {Object}

This method registers an event listener. The caller can specify whether the listener is to be invoked in the capture phase only or only in the target and bubbling phases.

Parameters:

type: {String}

A string providing the event type for which the listener is to be registered

callback: {Function}

The function to be called if the event occurs; the function receives the following argument:

- (Event) evt - a reference to the fired event. See [nokia.maps.dom.Event](#)

useCapture: {Boolean}

A flag that indicates if the event listener is to be used for the capture phase only (true - the listener is not invoked during the target and bubbling phases), or for target and bubbling phases (false)

Returns:

{Object} A reference to the given `nokia.maps.dom.EventTarget` object

addListenerNS(namespaceURI, type, callback, useCapture):

{[nokia.maps.dom.EventTarget](#)}

This method registers an event listener for events originating from a specific namespace. The caller can specify whether the listener is to be invoked in the capture phase only or only in the target and bubbling phases.

here

Parameters:

namespaceURI: {String}

A string specifying the `Event.namespaceURI` associated with the event for which the user is registering

type: {String}

A string specifying the event type for which the listener is to be registered

callback: {Function}

The function to be called if the event occurs; the function receives the following argument:

- (Event) evt - a reference to the fired event. See [nokia.maps.dom.Event](#)

useCapture: {Boolean}

A flag indicating if the event listener is to be used for the capture phase only (`true` - the listener is not invoked during the target and bubbling phases), or for target and bubbling phases (`false`)

Returns:

{[nokia.maps.dom.EventTarget](#)}

A reference to the given event target object

addListener(obj)

This method provides a means of registering multiple event listeners.

Parameters:

obj: {Object}

A hash map object where key is name of event and value is an array that contains callback function which should be called if the event occurs and boolean value that indicates if the event listener have to be used only for the capture phase

```
{
  "eventName": [callback, useCapture],
}
```

here

```
"eventName2": [callback, useCapture]
}
```

static **catchException**(enable)

This method enables exception catching in the event dispatcher to make it possible to ignore errors in event handlers or it disables exception catching in the event dispatcher for debugging purposes. By default, exceptions are caught and written to the debug console.

Parameters:

enable: {Boolean}

true to enable exception catching; false otherwise

disableDrag(): {*nokia.maps.dom.EventTarget*}

This method disables dragging of this event target, causing this event target not to receive dragstart, drag and dragend events.

Note that although enableDrag disables user selection, this method does not enable it.

Returns:

{*nokia.maps.dom.EventTarget*}

The event target itself

disableUserSelect(): {*nokia.maps.dom.EventTarget*}

This method prevents user selection of text and elements within the event target and prevents the magnifier on the iPhone or other similar mobile devices.

Note that if you wish to listen to the longpress event, you should disable the user selection, because otherwise the iPhone and other mobile devices may show a magnifier, making it harder for you to interpret the longpress event.

Returns:

{*nokia.maps.dom.EventTarget*}

The event target itself

here

dispatch(evt): {Boolean}

This method dispatches an event.

The default implementation depends on whether the event has a namespaceURI property set or not. If this property is not set, the default implementation searches for a parent object using the property parentNode.

If the event has namespaceURI set, then the property parentNodes is checked, which contains a hash table with "namespaceURI" as key and the parent object for this namespace as value. If no match is found using the "namespaceURI" mechanism, then the parentNode property is checked as an alternative parent.

Parameters:

evt: {*nokia.maps.dom.Event* | Object}

An object representing the event to be dispatched; it is either an instance of *nokia.maps.dom.Event* or a native browser event.

Returns:

{Boolean} A flag indicating whether any of the listeners which handled the event called *Event.preventDefault()*. If *Event.preventDefault()* was called, the return value is false, otherwise it is true

static **dispatchEvent**(target, event, dispatchPath): {Boolean}

This method dispatches an event to the given event target.

The default implementation depends on whether the event has a namespaceURI property set or not. If this property is not set, then the default implementation searches for a parent object using the property parentNode.

If the event has namespaceURI set, then the property parentNodes is checked, which contains a hash table with "namespaceURI" as key and the parent object for this namespace as value. If no match is found using the "namespaceURI" mechanism, then the parentNode property is checked as an alternative parent.

Parameters:

target: {*nokia.maps.dom.EventTarget*}

here

The event target to which the event is to be dispatched

event: {*nokia.maps.dom.Event* | Object}

The event to be dispatched; it is either a *nokia.maps.dom.Event* object or a native browser event

dispatchPath: {Node[]} [optional]

List of elements like from
nokia.maps.dom.EventTarget#getDispatchPath method

Returns:

{Boolean} A value indicating whether any of the listeners which have handled the event called *preventDefault*; if *preventDefault* was called false is returned, otherwise true

enableDrag(): {*nokia.maps.dom.EventTarget*}

This method enables dragging of this event target and disables the user selection, allowing this event target to receive *dragstart*, *drag* and *dragend* events.

Returns:

{*nokia.maps.dom.EventTarget*}

The event target itself

enableUserSelect(): {*nokia.maps.dom.EventTarget*}

This method allows the user to select text and elements within the event target and allows the magnifier on the iPhone or other similar mobile devices.

Returns:

{*nokia.maps.dom.EventTarget*}

The event target itself

hitTest(pageX, pageY): {Boolean}

here

This method tests if the given x/y position relative to the document lies within the outer bounding box of the node corresponding to the given `EventTarget` object.

If this object is not a DOM node, the method must be overloaded to ensure that `mouseleave` events are fired correctly.

Parameters:

<code>pageX:</code>	<code>{Number}</code>
	The horizontal position of the CSS pixel relative to the document to test.
<code>pageY:</code>	<code>{Number}</code>
	The vertical position of the CSS pixel relative to the document to test.

Returns:

<code>{Boolean}</code>	<code>true</code> if the pixel lies within the node; <code>false</code> otherwise.

`insertListener(type, callback, useCapture): {Object}`

This method registers an event listener as the first listener in the listener chain. The caller can specified whether the listener is to be registered for the capture phase only, or for target and bubbling phases.

Parameters:

<code>type:</code>	<code>{String}</code>
	Specifies the event type for which the listener is to be registered
<code>callback:</code>	<code>{Function}</code>
	The function to be called if the event occurs; the function receives the following argument: (Event) <code>evt</code> - a reference to the fired event. See nokia.maps.dom.Event
<code>useCapture:</code>	<code>{Boolean}</code>
	Indicates if the event listener is to be used for the capture phase only (<code>true</code> - the listener is not invoked during the target and bubbling phases), or for target and bubbling phases (<code>false</code>)

here

Returns:

{Object} A reference to the given `nokia.maps.dom.EventTarget` object

insertListenerNS(namespaceURI, type, callback, useCapture):
{[nokia.maps.dom.EventTarget](#)}

This method registers an event listener for events originating from a specific namespace as the first in the listener chain. The caller can specify whether the listener is to be invoked in the capture phase only or only in the target and bubbling phases.

Parameters:

namespaceURI: {String}

A string specifying the `Event.namespaceURI` associated with the event for which the listener is to be registered

type: {String}

A string specifying the event type for which the listener is to be registered

callback: {Function}

The function to be called if the event occurs; the function receives the following argument:

- (Event) evt - a reference to the fired event. See [nokia.maps.dom.Event](#)

useCapture: {Boolean}

A flag indicating if the event listener is to be used for the capture phase only (`true` - the listener is not invoked during the target and bubbling phases), or for target and bubbling phases (`false`)

Returns:

{[nokia.maps.dom.EventTarget](#)}

A reference to the given event target object

removeAllListeners()

This method remove all registered listeners from current `EventTarget`

here

removeListener(type, listener, useCapture): {*nokia.maps.dom.EventTarget*}

This method removes an event listener.

Parameters:

type: {String}

A string specifying the event type for which the listener was registered

listener: {Function}

The function to be removed

useCapture: {Boolean}

A flag indicating if the event listener was to be used for the capture phase only (`true` - the listener is not invoked during the target and bubbling phases), or for target and bubbling phases (`false`)

Returns:

{*nokia.maps.dom.EventTarget*}

A reference to the given event target object

removeListenerNS(namespaceURI, type, listener, useCapture):

{*nokia.maps.dom.EventTarget*}

This method removes an event listener for events originating from a specific namespace.

Parameters:

namespaceURI: {String}

A string specifying the `Event.namespaceURI` associated with the event for which the listener was registered

type: {String}

A string specifying the event type for which the listener was registered

listener: {Function}

here

The function to be removed

useCapture: {Boolean}

A flag indicating if the event listener was to be used for the capture phase only (true - the listener is not invoked during the target and bubbling phases), or for target and bubbling phases (false)

Returns:

{[nokia.maps.dom.EventTarget](#)}

A reference to the given event target object

Interface: FocusEvent

This interface is a member of [nokia.maps.dom](#).

Extends: [nokia.maps.dom.Event](#)

Interface Summary

This event is fired if a target receives or loses the focus.

[For full details, see [nokia.maps.dom.FocusEvent](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.dom.Event](#) :

[AT_TARGET](#), [bubbles](#), [BUBBLING_PHASE](#), [canBubble](#), [cancelable](#), [canSicker](#), [CAPTURING_PHASE](#), [currentTarget](#), [defaultPrevented](#), [eventPhase](#), [namespaceURI](#), [nativeEvent](#), [page](#), [propagation](#), [PROPAGATION_OK](#), [PROPAGATION_STOP](#), [PROPAGATION_STOP_IMMEDIATE](#), [target](#), [timeStamp](#), [type](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.dom.Event](#) :

[cancel](#), [clone](#), [preventDefault](#), [preventUnload](#), [stopImmediatePropagation](#), [stopPropagation](#)

here

Interface Description

Event is fired when a target receives or loses the focus.

Note that this class represents an unimplemented interface. It is included here to document its properties. The class is derived from [nokia.maps.dom.Event](#), but, in addition to the properties of the parent class, it has the properties described here.

Interface: FocusEventTarget

This interface is a member of [nokia.maps.dom](#).

Interface Summary

This class is a *virtual* interface that exists only for documentation purposes.

[For full details, see [nokia.maps.dom.FocusEventTarget](#)]

Table 89: Event Summary

Events
blur This event is fired if a target has lost the focus.
focus This event is fired when an event target has received the focus.
focusin This event is fired before a target receives the focus.
focusout This event is fired before a target is about to lose the focus.

Interface Description

Each class implementing this interface declares that it can act as the target for certain events. Each event in this interface (and therefore also within the classes implementing the interface) represents an event of a specific type.

The following example shows event handling for events of the type "click":

```
// Note that "obj" can be either a DOM node or any other JavaScript object.
var obj = nokia.maps.dom.EventTarget( {} );
obj.addListener("click", function (evt) {
  console.log("This is the '"+evt.type+"' event!");
});
```

here

```
obj.dispatch( new nokia.maps.dom.Event({
  type: "click"
}));
```

The example creates an instant of `nokia.maps.dom.EventTarget`, adds a listener for "click" events to it, and dispatches a "click" event to all registered listeners (in this case one) - the listeners receive the object that represents the target of the event.

For more information about dispatching events, please refer to the documentation of the [nokia.maps.dom.EventTarget](#).

Event Details

blur

This event is fired if a target has lost the focus. The event is dispatched after the target has lost the focus.

Event Handler Parameters:

evt	{ nokia.maps.dom.FocusEvent }
	An object representing the event

focus

This event is fired when an event target has received the focus. The event is dispatched after the target has received the focus.

Event Handler Parameters:

evt	{ nokia.maps.dom.FocusEvent }
	An object representing the event

focusin

This event is fired before a target receives the focus. The event is dispatched before the target receives the focus.

here

Event Handler Parameters:

evt {*nokia.maps.dom.FocusEvent*}

An object representing the event

focusout

This event is fired before a target is about to lose the focus. The event is dispatched before the target loses the focus.

Event Handler Parameters:

evt {*nokia.maps.dom.FocusEvent*}

An object representing the event

Interface: KeyboardEvent

This interface is a member of *nokia.maps.dom*.

Extends: *nokia.maps.dom.Event*

Interface Summary

This event is fired when a keyboard key is pressed or released.

[For full details, see *nokia.maps.dom.KeyboardEvent*]

Table 90: Property Summary

Properties	
altKey : {Boolean}	This property indicates if the Alt key modifier is activated (true) or not (false).
ctrlKey : {Boolean}	This property indicates if the control (Ctrl) was used as a key modifier (true) or not (false).
readonly DOM_KEY_LOCATION_JOYSTICK : {Number}	This field is an identifier indicating that the key activation originated on a game controller or a joystick on a mobile device.
readonly DOM_KEY_LOCATION_LEFT : {Number}	

here

Properties

This field is an identifier indicating that the activated key was a left-hand-side key (there is more than one possible location for this key).

readonly **`DOM_KEY_LOCATION_MOBILE`**: {Number}

This field is an identifier indicating that the key activation originated on a mobile device, either on a physical keypad or a virtual keyboard.

readonly **`DOM_KEY_LOCATION_NUMPAD`**: {Number}

This field is an identifier indicating that the key activation originated on the numeric keypad or with a virtual key corresponding to the numeric keypad.

readonly **`DOM_KEY_LOCATION_RIGHT`**: {Number}

This field is an identifier indicating that the the activated key was a right-hand-side key (there is more than one possible location for this key).

readonly **`DOM_KEY_LOCATION_STANDARD`**: {Number}

This field is an identifier indicating that a standard key was pressed, no distinction is made between the left or right version of the key, and the key did not originate from the numeric keypad (or did not originate from a virtual key corresponding to the numeric keypad).

`getModifierState`: {Boolean}

This method queries the state of a key modifier.

`keyIdentifier`: {String}

This property holds the identifier of the activated (pressed) key.

`keyLocation`: {Number}

This property indicates the location of the key that was pressed on the device, as described in Keyboard event types.

`metaKey`: {Boolean}

This property indicates if the Meta key modifier is activated (`true`) or not (`false`).

`repeat`: {Boolean}

This property indicates if a key has been pressed repeatedly or has been held down to generate repeated key strokes (`true`).

`shiftKey`: {Boolean}

This property indicates if the shift (Shift) key modifier is activated (`true`) or not (`false`).

Directly Inherited Properties

Inherited from class [*nokia.maps.dom.Event*](#) :

[*AT_TARGET*](#), [*bubbles*](#), [*BUBBLING_PHASE*](#), [*canBubble*](#), [*cancelable*](#), [*canSicker*](#), [*CAPTURING_PHASE*](#), [*currentTarget*](#), [*defaultPrevented*](#), [*eventPhase*](#), [*namespaceURI*](#), [*nativeEvent*](#), [*page*](#), [*propagation*](#), [*PROPAGATION_OK*](#), [*PROPAGATION_STOP*](#), [*PROPAGATION_STOP_IMMEDIATE*](#), [*target*](#), [*timeStamp*](#), [*type*](#)

here

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.dom.Event](#) :

[cancel](#), [clone](#), [preventDefault](#), [preventUnload](#), [stopImmediatePropagation](#), [stopPropagation](#)

Interface Description

This event is fired if a key on the keyboard is pressed or released.

Note that this class represents an unimplemented interface. It is included here to document its properties. The class is derived from [nokia.maps.dom.Event](#), but, in addition to the properties of the parent class, it has the properties described here.

Property Details

altKey: {Boolean}

This property indicates if the Alt key modifier is activated (true) or not (false).

ctrlKey: {Boolean}

This property indicates if the control (Ctrl) was used as a key modifier (true) or not (false).

readonly **DOM_KEY_LOCATION_JOYSTICK**: {Number}

This field is an identifier indicating that the key activation originated on a game controller or a joystick on a mobile device. Example: the 'DownLeft' key on a game controller.

readonly **DOM_KEY_LOCATION_LEFT**: {Number}

This field is an identifier indicating that the activated key was a left-hand-side key (there is more than one possible location for this key). Example: the left 'Control' key on a PC 101 Key US keyboard.

readonly **DOM_KEY_LOCATION_MOBILE**: {Number}

This field is an identifier indicating that the key activation originated on a mobile device, either on a physical keypad or a virtual keyboard. Example: the '#' key or softkey on a mobile device.

here

readonly **DOM_KEY_LOCATION_NUMPAD**: {Number}

This field is an identifier indicating that the key activation originated on the numeric keypad or with a virtual key corresponding to the numeric keypad. Example: the '1' key on a PC 101 Key US keyboard located on the numeric pad.

readonly **DOM_KEY_LOCATION_RIGHT**: {Number}

This field is an identifier indicating that the activated key was a right-hand-side key (there is more than one possible location for this key). Example: the right 'Shift' key on a PC 101 Key US keyboard.

readonly **DOM_KEY_LOCATION_STANDARD**: {Number}

This field is an identifier indicating that a standard key was pressed, no distinction is made between the left or right version of the key, and the key did not originate from the numeric keypad (or did not originate from a virtual key corresponding to the numeric keypad). Example: the 'Q' key on a PC 101 Key US keyboard.

getModifierState: {Boolean}

This method queries the state of a key modifier.

keyIdentifier: {String}

This property holds the identifier of the activated (pressed) key. The key identifiers are defined in the Key identifiers set. Implementations that are unable to identify a key must use the key identifier 'Unidentified'.

keyLocation: {Number}

This property indicates the location of the key that was pressed on the device, as described in Keyboard event types.

metaKey: {Boolean}

This property indicates if the Meta key modifier is activated (true) or not (false).

repeat: {Boolean}

here

This property indicates if a key has been pressed repeatedly or has been held down to generate repeated key strokes (`true`). Depending on the system configuration, holding down a key may result in multiple consecutive keydown events, keypress events, and textInput events, for appropriate keys. For mobile devices which have long-key-press behavior, the first key event with a repeat attribute value of `true` indicates a long-key-press. The length of time that the key must be pressed in order to begin repeating is configuration dependent.

shiftKey: {Boolean}

This property indicates if the shift (Shift) key modifier is activated (`true`) or not (`false`).

Interface: KeyboardEventTarget

This interface is a member of [nokia.maps.dom](#).

Interface Summary

This class is a *virtual* interface that does exist for documentation purposes.

[For full details, see [nokia.maps.dom.KeyboardEventTarget](#)]

Table 91: Event Summary

Events
keydown This event is fired if a key is pressed and delivered to the node that currently has the focus.
keyup This event is fired if a key is released and delivered to the node that currently has the focus.

Interface Description

Each class implementing this interface declares that it can act as the target for certain events. Each event in this interface (and therefore also within the classes implementing the interface) represents an event of a specific type.

The following example shows event handling for events of the type "click":

```
// Note that "obj" can be either a DOM node or any other JavaScript object.
var obj = nokia.maps.dom.EventTarget( {} );
obj.addListener("click", function (evt) {
  console.log("This is the '"+evt.type+"' event!");
});
```

here

```
obj.dispatch( new nokia.maps.dom.Event({
  type: "click"
}));
```

The example creates an instant of `nokia.maps.dom.EventTarget`, adds a listener for "click" events to it, and dispatches a "click" event to all registered listeners (in this case one) - the listeners receive the object that represents the target of the event.

For more information about dispatching events, please refer to the documentation of the [nokia.maps.dom.EventTarget](#).

Event Details

keydown

This event is fired if a key is pressed and delivered to the node that currently has the focus.

Event Handler Parameters:

evt	{ nokia.maps.dom.KeyboardEvent }
	An object representing the event

keyup

This event is fired if a key is released and delivered to the node that currently has the focus.

Event Handler Parameters:

evt	{ nokia.maps.dom.KeyboardEvent }
	An object representing the event

Interface: MouseEvent

This interface is a member of [nokia.maps.dom](#).

Extends: [nokia.maps.dom.Event](#)

Interface Summary

This event is fired for all mouse related events like moves, clicks, etc.

[For full details, see [nokia.maps.dom.MouseEvent](#)]

here

Table 92: Property Summary

Properties
altKey: {Boolean} This field indicates the activation state of the modifier key Alt.
button: {Number} This property holds the numeric id of a mouse button that has changed state, because it was pressed or released.
buttonState: {Number} This property holds the current state of all mouse buttons.
clientX: {Number} The horizontal coordinate at which the event occurred relative to the viewport associated with the event.
clientY: {Number} The vertical coordinate at which the event occurred relative to the viewport associated with the event.
ctrlKey: {Boolean} This field indicates the activation state of the modifier key Ctrl.
getModifierState: {Boolean} This method queries the state of a modifier on the basis of the key identifier supplied by the caller.
metaKey: {Boolean} This field indicates the activation state of the modifier key Meta.
pageX: {Number} The horizontal coordinate at which the event occurred relative to the document associated with the event.
pageY: {Number} The vertical coordinate at which the event occurred relative to the document associated with the event.
relatedTarget: {<i>nokia.maps.dom.EventTarget</i>} This property identifies a secondary event target related to a UI event, depending on the type of event: • mouseover event - the property refers to the target over which the mouse pointer or finger is has begun to pass or has passed, if available • mouseout event - the property contains the reference to the target which the mouse is leaving, if available • drag event - the property refers to the target that located below the mouse pointer or finger, if available • dragenter, dragleave or drop event - the property contains the reference to the event target that is currently being dragged (in other words, the target of the dragstart event) • dragend event - the property refers to the event target on which the dragged object has been dropped successfully; if the drop failed for any reason the property is null Therefore, this property only holds information relevant for mouseover, mouseout, drag, dragenter, dragleave, drop or dragend events; otherwise it is null.
screenX: {Number} The horizontal coordinate at which the event occurred relative to the origin of the screen coordinate system.

here

Properties

screenY: {Number}

The vertical coordinate at which the event occurred relative to the origin of the screen coordinate system.

shiftKey: {Boolean}

This field indicates the activation state of the modifier key Shift.

targetX: {Number}

The x-position of the cursor relative to the target.

targetY: {Number}

The y-position of the cursor relative to the target.

Directly Inherited Properties

Inherited from class [nokia.maps.dom.Event](#) :

[AT_TARGET](#), [bubbles](#), [BUBBLING_PHASE](#), [canBubble](#), [cancelable](#), [canSlicker](#), [CAPTURING_PHASE](#), [currentTarget](#), [defaultPrevented](#), [eventPhase](#), [namespaceURI](#), [nativeEvent](#), [page](#), [propagation](#), [PROPAGATION_OK](#), [PROPAGATION_STOP](#), [PROPAGATION_STOP_IMMEDIATE](#), [target](#), [timeStamp](#), [type](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.dom.Event](#) :

[cancel](#), [clone](#), [preventDefault](#), [preventUnload](#), [stopImmediatePropagation](#), [stopPropagation](#)

Interface Description

This class represents all mouse-related events.

Note that this class represents an unimplemented interface. It is included here to document its properties. The class is derived from [nokia.maps.dom.Event](#), but, in addition to the properties of the parent class, it has the properties described here.

Property Details

altKey: {Boolean}

This field indicates the activation state of the modifier key Alt. It is set to `true` if the modifier is activated.

here

button: {Number}

This property holds the numeric id of a mouse button that has changed state, because it was pressed or released.

- 0 - indicates the normal button of the mouse that is used to click on user interface buttons and menu items or to select text (in general this is the left mouse button or the one button on a Macintosh mouse)
- 2 - indicates the contextual property button of the mouse if present (normally, this means the right mouse button that is used to display a context menu)
- 1 - indicates the extra button (typically, the middle button, which is often combined with the mouse wheel)
- values higher than 2 - indicate other mouse buttons that may be present or simulated on some mouse devices

buttonState: {Number}

This property holds the current state of all mouse buttons.

This property is a bitmask where bit zero represents the state of the left mouse button, bit one the state of the middle mouse button, bit two the state of the right mouse button and bit three and higher the state of any further mouse buttons that may be present or simulated. This state is set for all events. You can detect if a specific button has been pressed, like this: `if (event.buttonState & (1 << button))` . . ., where `button` is the identifier as defined by the W3C (zero for the left button, one for the middle button, two for the right button and three, etc. for further buttons).

Note that within a "mouseup" event, the value of this property is up-to-date. For example, while the `button` property may be set to 1 to indicate that the middle button has been released, bit one in the `buttonState` property is already cleared as the button has already been released.

This is a proprietary member and not part of the W3C interface specification this class implements.

clientX: {Number}

The horizontal coordinate at which the event occurred relative to the viewport associated with the event.

clientY: {Number}

The vertical coordinate at which the event occurred relative to the viewport associated with the event.

here

ctrlKey: {Boolean}

This field indicates the activation state of the modifier key Ctrl. It is set to `true` if the modifier is activated.

getModifierState: {Boolean}

This method queries the state of a modifier on the basis of the key identifier supplied by the caller.

metaKey: {Boolean}

This field indicates the activation state of the modifier key Meta. It is set to `true` if the modifier is activated.

pageX: {Number}

The horizontal coordinate at which the event occurred relative to the document associated with the event.

This is a proprietary member and not part of the W3C interface specification this class implements.

pageY: {Number}

The vertical coordinate at which the event occurred relative to the document associated with the event.

This is a proprietary member and not part of the W3C interface specification this class implements.

relatedTarget: {[*nokia.maps.dom.EventTarget*](#)}

This property identifies a secondary event target related to a UI event, depending on the type of event:

- mouseover event - the property refers to the target over which the mouse pointer or finger is has begun to pass or has passed, if available
- mouseout event - the property contains the reference to the target which the mouse is leaving, if available
- drag event - the property refers to the target that located below the mouse pointer or finger, if available

here

- `dragenter`, `dragleave` or `drop` event - the property contains the reference to the event target that is currently being dragged (in other words, the target of the `dragstart` event)
- `dragend` event - the property refers to the event target on which the dragged object has been dropped successfully; if the drop failed for any reason the property is `null`

Therefore, this property only holds information relevant for `mouseover`, `mouseout`, `drag`, `dragenter`, `dragleave`, `drop` or `dragend` events; otherwise it is `null`.

Note that in a drag operation the property `relatedTarget` always contains the necessary counterpart of the event, but only for simulated events, not for native drag events.

screenX: {Number}

The horizontal coordinate at which the event occurred relative to the origin of the screen coordinate system.

screenY: {Number}

The vertical coordinate at which the event occurred relative to the origin of the screen coordinate system.

shiftKey: {Boolean}

This field indicates the activation state of the modifier key Shift. It is set to `true` if the modifier is activated.

targetX: {Number}

The x-position of the cursor relative to the target.

This is a proprietary member and not part of the W3C interface specification this class implements.

targetY: {Number}

The y-position of the cursor relative to the target.

This is a proprietary member and not part of the W3C interface specification this class implements.

Interface: `MouseEventTarget`

This interface is a member of nokia.maps.dom.

here

Interface Summary

This class is a *virtual* interface that does exist for documentation purposes.

[For full details, see [nokia.maps.dom.MouseEventTarget](https://api.nokia.maps.com/MouseEventTarget)]

Table 93: Event Summary

Events
<i>click</i> This event is fired after a mousedown/mouseup sequence at the same target has occurred.
<i>dblclick</i> This event is fired after two click events have been fired within a certain time period.
<i>longpress</i> This event is fired after a mouse button has been pressed for a certain amount of time without starting a drag.
<i>mousedown</i> This event is fired if a mouse button has been pressed.
<i>mouseenter</i> This event is fired if the mouse cursor enters the visible area of a node -- it is fired at the node that the mouse has entered.
<i>mouseleave</i> This event is fired when the mouse cursor leaves the physical area of a node -- it is fired at the node the mouse has left.
<i>mousemove</i> This event is fired if the mouse is moved.
<i>mouseout</i> This event is fired if the mouse cursor leaves the visible area of a node -- the event is fired at the node that the mouse has left.
<i>mouseover</i> This event is fired when the mouse cursor enters the visible area of a node -- the event is fired at the node that is entered.
<i>mouseup</i> This event is fired if a mouse button has been released.
<i>mousewheel</i> This event is fired if the mousewheel is moved.

here

Interface Description

Each class implementing this interface declares that it can act as the target for certain events. Each event in this interface (and therefore also within the classes implementing the interface) represents an event of a specific type.

The following example shows event handling for events of the type "click":

```
// Note that "obj" can be either a DOM node or any other JavaScript object.
var obj = nokia.maps.dom.EventTarget( {} );
obj.addListener("click", function (evt) {
  console.log("This is the '"+evt.type+"' event!");
});

obj.dispatch( new nokia.maps.dom.Event({
  type: "click"
}));
```

The example creates an instant of `nokia.maps.dom.EventTarget`, adds a listener for "click" events to it, and dispatches a "click" event to all registered listeners (in this case one) - the listeners receive the object that represents the target of the event.

For more information about dispatching events, please refer to the documentation of the [nokia.maps.dom.EventTarget](#).

Event Details

click

This event is fired after a mousedown/mouseup sequence at the same target has occurred.

Event Handler Parameters:

evt	{ nokia.maps.dom.MouseEvent }
	An object representing the event

dblclick

This event is fired after two click events have been fired within a certain time period.

Event Handler Parameters:

evt	{ nokia.maps.dom.MouseEvent }

here

An object representing the event

longpress

This event is fired after a mouse button has been pressed for a certain amount of time without starting a drag.

Event Handler Parameters:

evt {*nokia.maps.dom.MouseEvent*}

An object representing the event

mousedown

This event is fired if a mouse button has been pressed.

Event Handler Parameters:

evt {*nokia.maps.dom.MouseEvent*}

An object representing the event

mouseenter

This event is fired if the mouse cursor enters the visible area of a node -- it is fired at the node that the mouse has entered.

The difference between the `mouseenter` and `mouseleave` events compared to the `mouseover` and `mouseout` events is that there can be multiple `mouseenter` events without even one `mouseleave` event. This happens, for example, if you have a `<div>` with the size of 100 x 100 pixels that contains a centered image measuring 50 x 50 pixels. When the mouse cursor enters the outer `<div>`, the `<div>` receives a `mouseover` and a `mouseenter` event (note that the order is not guaranteed). However, if the mouse cursor now moves over the image, the outer `<div>` receives a `mouseout` event, because the mouse is no longer within its visual area, and now the image gets a `mouseover` event. Additionally, the image receives a `mouseenter` event, but the outer `<div>` does not get a `mouseleave` event, because the mouse has not left the physical area of the outer `<div>`.

here

The `mouseenter` and `mouseleave` events are very useful, for example, when building tooltips that are to appear below the mouse cursor or when showing overlay information as long as the mouse cursor is above a specific area that contains child nodes.

Event Handler Parameters:

`evt` `{nokia.maps.dom.MouseEvent}`

An object representing the event

mouseleave

This event is fired when the mouse cursor leaves the physical area of a node -- it is fired at the node the mouse has left.

The difference between the `mouseenter` and `mouseleave` events compared to the `mouseover` and `mouseout` events is that there can be multiple `mouseenter` events without even one `mouseleave` event. This happens, for example, if you have a `<div>` with the size of 100 x 100 pixels that contains a centered image measuring 50 x 50 pixels. When the mouse cursor enters the outer `<div>`, the `<div>` receives a `mouseover` and a `mouseenter` event (note that the order is not guaranteed). However, if the mouse cursor now moves over the image, the outer `<div>` receives a `mouseout` event, because the mouse is no longer within its visual area, and now the image gets a `mouseover` event. Additionally, the image receives a `mouseenter` event, but the outer `<div>` does not get a `mouseleave` event, because the mouse has not left the physical area of the outer `<div>`.

The `mouseenter` and `mouseleave` events are very useful, for example, when building tooltips that are to appear below the mouse cursor or when showing overlay information as long as the mouse cursor is above a specific area that contains child nodes.

Event Handler Parameters:

`evt` `{nokia.maps.dom.MouseEvent}`

An object representing the event

mousemove

This event is fired if the mouse is moved.

here

Event Handler Parameters:

evt {*nokia.maps.dom.MouseEvent*}

An object representing the event

mouseout

This event is fired if the mouse cursor leaves the visible area of a node -- the event is fired at the node that the mouse has left.

Event Handler Parameters:

evt {*nokia.maps.dom.MouseEvent*}

An object representing the event

mouseover

This event is fired when the mouse cursor enters the visible area of a node -- the event is fired at the node that is entered.

Event Handler Parameters:

evt {*nokia.maps.dom.MouseEvent*}

An object representing the event

mouseup

This event is fired if a mouse button has been released.

Event Handler Parameters:

evt {*nokia.maps.dom.MouseEvent*}

An object representing the event

here

mousewheel

This event is fired if the mousewheel is moved.

Event Handler Parameters:

evt {[nokia.maps.dom.WheelEvent](#)}

An object representing the event

Class: Page

This class is a member of [nokia.maps.dom](#).

Class Summary

This class defines a mixin which extends given DOM node.

[For full details, see [nokia.maps.dom.Page](#)]

Table 94: Property Summary

Properties	
readonly body : {Element}	This property holds a reference to the body element of the document of this page.
static browser :	This static object reflects details of the current browser and user interface.
readonly document : {document}	This property holds a reference to the document that is rendered into the window referenced by window and to which the given instance of the page class is bound.
readonly documentMode : {Number}	This property is undefined unless the current browser is Microsoft Internet Explorer.
static DOUBLE_CLICK_TIME : {Number}	The time in milliseconds between two "click" events that raises a "dblclick" event.
static DOUBLE_TAP_TIME : {Number}	The time in milliseconds between two "tap" events that raises a "dbltap" event.
readonly head : {Element}	This property holds a reference to the head element of the document of this page.
readonly height : {Number}	

here

Properties

This property indicates the height of the document in CSS pixels.

readonly **html**: {Element}

This property holds a reference to the HTML element of the document of this page.

readonly **layoutHeight**: {Number}

This property indicates the layout height, which is space measured in CSS pixels into which the document is rendered, without scrollbars.

readonly **layoutWidth**: {Number}

This property indicates the layout width as space measured in CSS pixels into which the document is rendered, without scrollbars.

static **LONGPRESS_INTERVAL**: {Number}

The interval in milliseconds after which a "longpress" event is fired if a button is pressed down.

static **MOUSEMOVE_THRESHOLD**: {Number}

The minimum delay (in milliseconds) after a mousemove event has been fired during which no further mousemove events are fired.

readonly **orientation**: {Number}

This property holds the current orientation of the layout viewport (not of the document).

static **platform**:

This static object holds information about the current platform (operating system, mobile device, etc).

readonly **quirksMode**: {Boolean}

This property is set to true if the document is rendered in quirks mode; false if rendered in CSS compatibility mode.

static **RESIZE_IDL_TIME**: {Number}

A number of milliseconds that a node must be not resized before the "resizeend" event is fired.

static **RESIZE_SLEEP_MAX_TIME**: {Number}

The maximum time in milliseconds to wait until the watched DOM nodes are checked again for a resize.

static **RESIZE_SLEEP_MIN_TIME**: {Number}

The minimum time in milliseconds to wait after a "resizestart" event before the watched DOM nodes are re-checked (to determine whether the resize has been completed and the "resizeend" event can be fired).

readonly **scrollbarsHeight**: {Number}

This property indicates the height of a vertical scrollbar in device pixels.

readonly **scrollbarsWidth**: {Number}

This property indicates the width of a vertical scrollbar in device pixels.

readonly **viewportHeight**: {Number}

This property indicates the viewport height, which is space available to render the layout measured in device pixels.

readonly **viewportWidth**: {Number}

here

Properties

This property indicates the viewport width, which is space available to render the layout measured in device pixels.

readonly **width**: {Number}

This property indicates the width of the document in CSS pixels.

readonly **window**: {window}

This property holds a reference to the window in which the document is rendered.

Table 95: Method Summary

Methods

altKeyDown () : {Boolean}

This method checks if the Alt key is currently pressed.

createElement (tagName, style) : {Node}

This method creates a new DOM node and sets the style properties to use the values supplied by the caller.

ctrlKeyDown () : {Boolean}

This method checks if the Ctrl key is currently pressed.

getClientRect (node) : {Object}

This method calculates the absolute position of a node relative to the document.

metaKeyDown () : {Boolean}

This method checks if the Meta key is currently pressed.

static **parseEvent** (e) : {*nokia.maps.dom.Event*}

This method parses the given native event and returns a corresponding normalized event object.

setStyle (node, style) : {Node}

This method copies style properties into the style of the given DOM node.

shiftKeyDown () : {Boolean}

This method checks if the Shift key is currently pressed.

update () : {*nokia.maps.dom.Page*}

This method updates all volatile properties of the page (e.g.

Class Description

This class defines a mixin that provides a common API for documents. To get an object that implements the page behavior, simply pass a document or a DOM node to the constructor as shown in the examples below:

```
// Query page support for the document.
```

here

```
var page = nokia.maps.dom.Page(document);  
  
// Query page support for the document of a DOM node.  
var node = document.getElementById("foo");  
page = nokia.maps.dom.Page(node);
```

The mixin defines a large number of helper methods and enables *nokia.maps.dom*-aligned event handling for all DOM nodes attached to the document. Therefore, we recommend that you get a Page object for any document whose nodes are to be event targets (*nokia.maps.dom.EventTarget*) at least once.

Note that unlike *nokia.maps.dom.EventTarget*, Page does not modify DOM nodes to which it is applied and attaches only one property to the document (`$jslPage`). This is why you must obtain an object implementing Page at least once for every document or DOM node.

Obtaining a Page object is very fast, except the first time, when the object must be created and bound to the document, which means that certain listeners must be registered with the document and some methods are created internally.

Details

The DOM package facilitates cross-browser event handling, simplifies the creation of custom events, offers access to information about the browser and the operating system, and provides a set of helper methods.

The package consists of the four main classes *nokia.maps.dom.Page*, *nokia.maps.dom.Event*, *nokia.maps.dom.EventTarget* and *nokia.maps.dom.DataTransfer*. *nokia.maps.dom.Page* can be considered an entry-point class here. It offers an abstraction of native browser events and most of the functionality in the package. The other classes are chiefly helper classes. The package offers cross-browser W3C-standards-compliant event handling and support for de facto standards for touch and gesture events.

The supported events and their meaning are documented within the different abstract pseudo event target classes:

- *nokia.maps.dom.TouchEventTarget*
- *nokia.maps.dom.DragEventTarget*
- *nokia.maps.dom.MouseEventTarget*
- *nokia.maps.dom.KeyboardEventTarget*
- *nokia.maps.dom.FocusEventTarget*

Note that these interfaces are defined here for documentation purposes only. They are referred to in the documentation for specific classes to show which events instances of those classes support. However, the design of event handling makes it possible for an object to receive additional undocumented events.

here

Event handling

Event handling itself is very simple and mostly W3C compatible, but it must be activated for native DOM nodes before it can be used. The reason is to prevent collisions with other frameworks that may be used together with this API. To add support for *aligned events* to a DOM node the interface [nokia.maps.dom.EventTarget](#) must be applied to the DOM node. This can be done simply by casting the DOM node into an [nokia.maps.dom.EventTarget](#) as the following example shows:

```
// Create a few shortcuts.
var Page = nokia.maps.dom.Page,
    EventTarget = nokia.maps.dom.EventTarget;

// Query page support for the document. This is very important, but needs
// only
// to be done once.
Page(document);

// Attach EventTarget interface to the document to allow aligned events in
// the
// document.
EventTarget(document);

// Add a listener for the click event to the document and show an alert,
// if the document is clicked.
document.addEventListener("click", function (evt) {
    alert(evt.type);
});

// Remove the listener so that an alert is shown only once.
document.removeEventListener("click", arguments.callee, false);
}, false);
```

Most of the events are quite selfexplanatory and the syntax to register or unregister listeners is similar to the W3C standard methods `addEventListener`, `removeEventListener` and `dispatchEvent`, except that the aligned methods are named `addListener`, `removeListener` and `dispatch` (these names do not contain the word "event").

Constructor Details

nokia.maps.dom.Page(element)

The constructor mixes the functionality of `Page` into the argument element.

Parameters:

element: {HTMLElement}

- the document or element to be extended with `Page` functionality

here

Property Details

readonly **body**: {Element}

This property holds a reference to the body element of the document of this page.

static **browser**:

This static object reflects details of the current browser and user interface. The object includes the following fields -- if one of these fields is set, the corresponding browser/interface has been detected:

- {String} **version** - this property holds the version number of the current browser as a string convertible into a number (e.g. "4.0")
- {Number} **realVersion** - this property holds the real major version of the browser; for IE this property may hold a different value from **version**, because when, for example, IE 9 simulates IE 8, **realVersion** contains "9", while **version** is "8.0"; **realVersion** can be used to work around a bug in the VML implementation of IE 8 which is not present in IE 9 (even if it runs in IE 8 compatibility mode)
- {String} **fullVersion** - this property holds the full version number of the current browser as a string which is not convertible to a number (e.g. "4.0.1")
- {String} **engineVersion** - this property holds the version number of the rendering engine of the browser as a string
- {Boolean} **chrome** - this property indicates if the current browser is Chrome
- {Boolean} **safari** - this property indicates if the current browser is Safari
- {Boolean} **opera** - this property indicates if the current browser is Opera
- {Boolean} **msie** - this property indicates if the current browser is Internet Explorer
- {Boolean} **mozilla** - this property indicates if the current browser is a Mozilla browser
- {Boolean} **konqueror** - this property indicates if the current browser is a Konqueror browser
- {Boolean} **camino** - this property indicates if the current browser is a Camino browser
- {Boolean} **webkit** - this property indicates if the current browser uses the WebKit rendering engine
- {Boolean} **gecko** - this property indicates if the current browser uses the Gecko rendering engine
- {Boolean} **dnd** - this property indicates if the browser supports native HTML 5 drag and drop
- {Boolean} **touch** - this property indicates if the touch interface is in use
- {Boolean} **mobile** - this property indicates if a mobile web browser is in use
- {Boolean} **web** - this property indicates if a desktop web browser is in use

here

- `{String} language` - this property holds the language of the browser as a string, for example "en-GB"

For example, to check if the current browser is from the Mozilla family, you can use the following code:

```
if (nokia.maps.dom.Page.browser.mozilla) {  
  // code applicable to Mozilla here  
} else {  
  // Code for non-Mozilla browsers here  
}
```

readonly **document**: {document}

This property holds a reference to the document that is rendered into the window referenced by window and to which the given instance of the page class is bound.

readonly **documentMode**: {Number}

This property is undefined unless the current browser is Microsoft Internet Explorer. In that case it contains the document mode in which the document of this page is rendered, with the possible values of 5 (quirks mode, IE 5 compatible), 7 (IE 6/7 standard compliant mode), 8 (IE 8 standard compliant mode) or 9 (IE 9 HTML 5 mode).

static **DOUBLE_CLICK_TIME**: {Number}

The time in milliseconds between two "click" events that raises a "dblclick" event. This property is used only for browsers that do not fire native double click events or where there are no double click events under specific circumstances (browser bugs). For example, Chrome does not fire a right mouse button double-click event at all.

static **DOUBLE_TAP_TIME**: {Number}

The time in milliseconds between two "tap" events that raises a "dbltap" event.

readonly **head**: {Element}

This property holds a reference to the head element of the document of this page.

here

readonly **height**: {Number}

This property indicates the height of the document in CSS pixels. It is available in all browsers.

readonly **html**: {Element}

This property holds a reference to the HTML element of the document of this page.

readonly **layoutHeight**: {Number}

This property indicates the layout height, which is space measured in CSS pixels into which the document is rendered, without scrollbars.

readonly **layoutWidth**: {Number}

This property indicates the layout width as space measured in CSS pixels into which the document is rendered, without scrollbars.

static **LONGPRESS_INTERVAL**: {Number}

The interval in milliseconds after which a "longpress" event is fired if a button is pressed down.

static **MOUSEMOVE_THRESHOLD**: {Number}

The minimum delay (in milliseconds) after a mousemove event has been fired during which no further mousemove events are fired. If set to zero or less mousemove event will be fired in near time (so as soon as possible after they occurred, but asynchronously).

readonly **orientation**: {Number}

This property holds the current orientation of the layout viewport (not of the document).

The rotation is in degrees clockwise: a value of 0 indicates portrait mode with a normal device screen; 90 indicates landscape mode, with the device screen rotated 90 degrees clockwise; 180 is portrait mode, with the device screen rotated 180 degrees clockwise; and finally 270 degrees means the device screen is in landscape mode with the screen rotated 270 degrees clockwise.

Note that these values differ from those reported natively (for example by the iPhone), but they are easier to handle for developers as they refer simply the clockwise rotation.

here

static **platform**:

This static object holds information about the current platform (operating system, mobile device, etc.) The object includes the following properties:

- {Boolean} windows - this property indicates if the current operating system is Windows
- {Boolean} mac - this property indicates if the current platform/operating system is Mac OS X
- {Boolean} linux - this property indicates if the current operating system is Linux (but not Maemo)
- {Boolean} maemo - this property indicates if the current operating system is Maemo
- {Boolean} meego - this property indicates if the current platform is Meego
- {Boolean} android - this property indicates if the current platform is Android
- {Boolean} iphone - this property indicates if the current platform is iPhone touch device
- {Boolean} ipad - this property indicates if the current platform is iPad touch device
- {Boolean} windowsphone - this property indicates if the current platform is Windows Phone device

For example, to check if the operating system is Linux, you can use the following code:

```
if (nokia.maps.dom.Page.platform.linux) {
  // code applicable to Linux here
} else {
  // non-Linux code here
}
```

readonly **quirksMode**: {Boolean}

This property is set to true if the document is rendered in quirks mode; false if rendered in CSS compatibility mode.

static **RESIZE_IDL_TIME**: {Number}

A number of milliseconds that a node must be not resized before the "resizeend" event is fired.

static **RESIZE_SLEEP_MAX_TIME**: {Number}

The maximum time in milliseconds to wait until the watched DOM nodes are checked again for a resize.

here

static **RESIZE_SLEEP_MIN_TIME**: {Number}

The minimum time in milliseconds to wait after a "resizestart" event before the watched DOM nodes are re-checked (to determine whether the resize has been completed and the "resizeend" event can be fired).

readonly **scrollbarsHeight**: {Number}

This property indicates the height of a vertical scrollbar in device pixels.

readonly **scrollbarsWidth**: {Number}

This property indicates the width of a vertical scrollbar in device pixels.

readonly **viewportHeight**: {Number}

This property indicates the viewport height, which is space available to render the layout measured in device pixels.

readonly **viewportWidth**: {Number}

This property indicates the viewport width, which is space available to render the layout measured in device pixels.

readonly **width**: {Number}

This property indicates the width of the document in CSS pixels. The property is valid only for WebKit-based browsers like Safari and Chrome. If the width of the document cannot be detected the value is null.

readonly **window**: {window}

This property holds a reference to the window in which the document is rendered.

Method Details

altKeyDown(): {Boolean}

This method checks if the Alt key is currently pressed.

here

Note that the information returned by this method may be inaccurate, because key-presses cannot be detected if the user changes the focus into a different window or browser tab.

Returns:

{Boolean} true if the Alt key is currently pressed; false otherwise

createElement(tagName, style): {Node}

This method creates a new DOM node and sets the style properties to use the values supplied by the caller.

```
var page = nokia.maps.dom.Page(document);
var node = page.createElement("DIV", {
  backgroundColor: "rgb(255,0,0)",
  width: "200px",
  height: "200px"
});
```

It is also possible to create a node with specific attributes and style:

```
var page = nokia.maps.dom.Page(document);
var node = page.createElement({
  tagName: "CANVAS",
  width: 200,
  height: 200,
  style: {
    position: "absolute",
    top: 0,
    left: 0,
    width: "200px",
    height: "200px"
  }
});
```

Parameters:

tagName: {String | Object}

The tag name of the DOM node to be created or an object, where the key `tagName` contains the tag name of the DOM node to be created and other keys reflect other properties of the node; the property `style` is used to declare the style sheet, however, if the `style` argument (see below) is supplied as well, it overwrites the style defined in `tagName`

style: {Object} [optional]

here

Returns:

ctrlKeyDown(): {Boolean}

Note that the information returned by this method may be inaccurate, because key-presses cannot be detected if the user changes the focus into a different window or browser tab.

```
{Boolean}      true if the control key is currently pressed; false otherwise
```

This method calculates the absolute position of a node relative to the document. The returned value can be compared with the `pageX` and `pageY` properties of mouse events.

node: {Node}

Returns:

<code>{Object}</code>	The position of the node within the document as an object containing the properties <code>left</code> , <code>top</code> , <code>right</code> , <code>bottom</code> , <code>width</code> and <code>height</code> , or <code>null</code> if detection fails or the given node is not a valid DOM node

Note that the information returned by this method may be inaccurate, because key-presses cannot be detected if the user changes the focus into a different window or browser tab.

Returns:

{Boolean} true if the Meta key is currently pressed; false otherwise

static **parseEvent**(e): {*nokia.maps.dom.Event*}

This method parses the given native event and returns a corresponding normalized event object.

Parameters:

e: {Object}
 The browser's native event object

Returns:

{*nokia.maps.dom.Event*}
 A normalized DOM level 3 event object

Example:

```
// If this is the listener to a native click event.
(function (nativeEvent) {
    // Fix the native event so that evt.button will complies to
    W3C standard.
    var evt = nokia.maps.dom.Page.parseEvent(nativeEvent);
    alert(evt.button == 0 ? "left" : (evt.button == 1 ?
    "middle" : "right"));
});
```

setStyle(node, style): {Node}

This method copies style properties into the style of the given DOM node.

```
var page = nokia.maps.dom.Page(document);
var node = document.createElement("DIV");
page.setStyle(node, {
    backgroundColor: "rgb(255,0,0)",
    width: "200px",
    height: "200px"
});
```

Note that the above example could have been done more efficient by directly using the `createElement` method from the `page` interface.

Parameters:

here

node: {Node}
The DOM node for which the style properties should be set.

style: {Object}
The style properties to be set in the node.

Returns:
{Node} The node.

shiftKeyDown(): {Boolean}

This method checks if the Shift key is currently pressed.

Note that the information returned by this method may be inaccurate, because key-presses cannot be detected if the user changes the focus into a different window or browser tab.

Returns:
{Boolean} true if the left or right Shift key is currently pressed; false otherwise

update(): {*nokia.maps.dom.Page*}

This method updates all volatile properties of the page (e.g., the document size).

Returns:
{*nokia.maps.dom.Page*}
Returns the page object itself (this).

Interface: TextEvent

This interface is a member of *nokia.maps.dom*.

Extends: *nokia.maps.dom.Event*

Interface Summary

This event is fired when text is entered.

[For full details, see *nokia.maps.dom.TextEvent*]

here

Table 96: Property Summary

Properties
<code>data</code> : {String} This property holds the data associated with the text event.
readonly <code>DOM_INPUT_METHOD_DROP</code> : {Number} This field is an identifier indicating that text was inserted as part of a drag-and-drop operation.
readonly <code>DOM_INPUT_METHOD_HANDWRITING</code> : {Number} This field is an identifier indicating that text was entered through a pen/tablet device and processed by handwriting recognition software.
readonly <code>DOM_INPUT_METHOD_IME</code> : {Number} This field is an identifier indicating that text was entered through an Input Method Editor.
readonly <code>DOM_INPUT_METHOD_KEYBOARD</code> : {Number} This field is an identifier indicating that text was entered, using through the keyboard.
readonly <code>DOM_INPUT_METHOD_MULTIMODAL</code> : {Number} This field is an identifier indicating that text was inserted as part of an operation involving multiple input modalities in combination, such as pointer-enhanced speech.
readonly <code>DOM_INPUT_METHOD_OPTION</code> : {Number} This field is an identifier indicating that text was selected from a set of options presented to the user, such as from a form.
readonly <code>DOM_INPUT_METHOD_PASTE</code> : {Number} This field is an identifier indicating that text was pasted in from a clipboard.
readonly <code>DOM_INPUT_METHOD_SCRIPT</code> : {Number} This field is an identifier indicating that text was inserted via a script operation on the DOM.
readonly <code>DOM_INPUT_METHOD_UNKNOWN</code> : {Number} This field is an identifier indicating that text was entered by an unknown method.
readonly <code>DOM_INPUT_METHOD_VOICE</code> : {Number} This field is an identifier indicating that text was input by a voice device and interpreted by speech recognition software.
<code>inputMode</code> : {Number} This property indicates the origin of the text input.

Directly Inherited Properties

Inherited from class *nokia.maps.dom.Event* :

AT_TARGET, *bubbles*, *BUBBLING_PHASE*, *canBubble*, *cancelable*, *canSicker*, *CAPTURING_PHASE*, *currentTarget*, *defaultPrevented*, *eventPhase*, *namespaceURI*, *nativeEvent*, *page*, *propagation*,

here

PROPAGATION_OK, PROPAGATION_STOP, PROPAGATION_STOP_IMMEDIATE, target, timeStamp, type

Method Summary

Directly Inherited Methods

Inherited from class *nokia.maps.dom.Event* :

cancel, clone, preventDefault, preventUnload, stopImmediatePropagation, stopPropagation

Interface Description

This event is fired if some text is entered from the keyboard, clipboard or from some other source.

Note that this class represents an unimplemented interface. It is included here to document its properties. The class is derived from *nokia.maps.dom.Event*, but, in addition to the properties of the parent class, it has the properties described here.

Property Details

data: {String}

This property holds the data associated with the text event. Its value are the characters generated by the character device. This may be a single Unicode character or a non-empty sequence of Unicode characters [Unicode]. Characters should be normalized as defined by the Unicode normalization form NFC. The property cannot be null or contain an empty string.

readonly **DOM_INPUT_METHOD_DROP**: {Number}

This field is an identifier indicating that text was inserted as part of a drag-and-drop operation. This may be associated with a drop event (described elsewhere).

readonly **DOM_INPUT_METHOD_HANDWRITING**: {Number}

This field is an identifier indicating that text was entered through a pen/tablet device and processed by handwriting recognition software. This may be associated with a specific event defined elsewhere.

readonly **DOM_INPUT_METHOD_IME**: {Number}

This field is an identifier indicating that text was entered through an Input Method Editor. This may be associated with a composition end event.

here

`readonly DOM_INPUT_METHOD_KEYBOARD: {Number}`

This field is an identifier indicating that text was entered, using through the keyboard. This may be associated with one or more of keypress, keydown, or keyup events.

`readonly DOM_INPUT_METHOD_MULTIMODAL: {Number}`

This field is an identifier indicating that text was inserted as part of an operation involving multiple input modalities in combination, such as pointer-enhanced speech. This may be associated with various other events (described elsewhere).

`readonly DOM_INPUT_METHOD_OPTION: {Number}`

This field is an identifier indicating that text was selected from a set of options presented to the user, such as from a form. This may be associated with various other events (described elsewhere).

`readonly DOM_INPUT_METHOD_PASTE: {Number}`

This field is an identifier indicating that text was pasted in from a clipboard. This may be associated with a paste event (described elsewhere).

`readonly DOM_INPUT_METHOD_SCRIPT: {Number}`

This field is an identifier indicating that text was inserted via a script operation on the DOM. This may be associated with one or more mutation events.

`readonly DOM_INPUT_METHOD_UNKNOWN: {Number}`

This field is an identifier indicating that text was entered by an unknown method.

`readonly DOM_INPUT_METHOD_VOICE: {Number}`

This field is an identifier indicating that text was input by a voice device and interpreted by speech recognition software. This may be associated with a specific event defined elsewhere.

`inputMode: {Number}`

here

This property indicates the origin of the text input. Its value is one of the input method identifiers defined on this class, for example, `DOM_INPUT_METHOD_UNKNOWN`, `DOM_INPUT_METHOD_KEYBOARD`, `DOM_INPUT_METHOD_PASTE`, etc.

Interface: Touch

This interface is a member of [nokia.maps.dom](#).

Interface Summary

The Touch class represents a single touch on the surface of the screen.

[For full details, see [nokia.maps.dom.Touch](#)]

Table 97: Property Summary

Properties
clientX : {Number} This property holds the x-coordinate of the touch location relative to the window viewport.
clientY : {Number} This property holds the y-coordinate of the touch location relative to the window viewport.
identifier : {Number} This property holds a unique identifier for this touch object.
pageX : {Number} This property holds the x-coordinate of the touch location in page coordinates.
pageY : {Number} This property holds the y-coordinate of the touch location in page coordinates.
screenX : {Number} This property holds the x-coordinate of the touch location in screen coordinates.
screenY : {Number} This property holds the y-coordinate of the touch location in screen coordinates.
target : {Number} This property holds the target of the given touch object.

Interface Description

A touch is the presence or movement of a finger that is part of a unique multi-touch sequence. Use the property `nokia.maps.dom.Event#changedTouches` to get all the touch objects that have changed in a touch event.

here

Property Details

clientX: {Number}

This property holds the x-coordinate of the touch location relative to the window viewport.

clientY: {Number}

This property holds the y-coordinate of the touch location relative to the window viewport.

identifier: {Number}

This property holds a unique identifier for this touch object.

pageX: {Number}

This property holds the x-coordinate of the touch location in page coordinates.

pageY: {Number}

This property holds the y-coordinate of the touch location in page coordinates.

screenX: {Number}

This property holds the x-coordinate of the touch location in screen coordinates.

screenY: {Number}

This property holds the y-coordinate of the touch location in screen coordinates.

target: {Number}

This property holds the target of the given touch object.

Interface: TouchEvent

This interface is a member of [*nokia.maps.dom*](#).

Extends: [*nokia.maps.dom.Event*](#)

here

Interface Summary

This event represents touches on touch screens.

[For full details, see [nokia.maps.dom.TouchEvent](#)]

Table 98: Property Summary

Properties
changedTouches: {nokia.maps.dom.Touch[]} This property holds a collection of nokia.maps.dom.Touch objects representing all touches that changed in this event.
rotation: {Number} This property holds the delta rotation since the start of an event in degrees, where clockwise is positive and counter-clockwise is negative.
scale: {Number} This property holds the distance between two fingers since the start of an event, as a multiplier of the initial distance.
targetTouches: {nokia.maps.dom.Touch[]} This property holds a collection of nokia.maps.dom.Touch objects representing all touches that have the same target as the original target of the first touch on the screen.
touches: {nokia.maps.dom.Touch[]} This property is an array holding all current instances of nokia.maps.dom.Touch that represent user's touches on the screen surface.

Directly Inherited Properties

Inherited from class [nokia.maps.dom.Event](#) :

[AT_TARGET](#), [bubbles](#), [BUBBLING_PHASE](#), [canBubble](#), [cancelable](#), [canSicker](#), [CAPTURING_PHASE](#), [currentTarget](#), [defaultPrevented](#), [eventPhase](#), [namespaceURI](#), [nativeEvent](#), [page](#), [propagation](#), [PROPAGATION_OK](#), [PROPAGATION_STOP](#), [PROPAGATION_STOP_IMMEDIATE](#), [target](#), [timeStamp](#), [type](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.dom.Event](#) :

[cancel](#), [clone](#), [preventDefault](#), [preventUnload](#), [stopImmediatePropagation](#), [stopPropagation](#)

here

Interface Description

This interface represents events relating to the user's touching, tapping or dragging on a touch screen, for example, on mobile phones or some computers (such as laptops) that support touch screens.

Note that this class represents an unimplemented interface. It is included here to document its properties. The class is derived from [nokia.maps.dom.Event](#), but, in addition to the properties of the parent class, it has the properties described here.

Property Details

changedTouches: {[nokia.maps.dom.Touch](#)[]}

This property holds a collection of [nokia.maps.dom.Touch](#) objects representing all touches that changed in this event.

rotation: {Number}

This property holds the delta rotation since the start of an event in degrees, where clockwise is positive and counter-clockwise is negative. The initial value is 0.0.

scale: {Number}

This property holds the distance between two fingers since the start of an event, as a multiplier of the initial distance. The initial value is 1.0. If the value is less than 1.0, the gesture is pinch close (to zoom out). If the value is greater than 1.0, the gesture is pinch open (to zoom in).

targetTouches: {[nokia.maps.dom.Touch](#)[]}

This property holds a collection of [nokia.maps.dom.Touch](#) objects representing all touches that have the same target as the original target of the first touch on the screen.

touches: {[nokia.maps.dom.Touch](#)[]}

This property is an array holding all current instances of [nokia.maps.dom.Touch](#) that represent user's touches on the screen surface.

Interface: TouchEventTarget

This interface is a member of [nokia.maps.dom](#).

here

Interface Summary

This class is a *virtual* interface that exists only for documentation purposes.

[For full details, see [nokia.maps.dom.TouchEventTarget](#)]

Table 99: Event Summary

Events
dblTap This event is fired after a two tap events have been fired in a certain amount of time.
gesturechange This high-level event is fired whenever either of the two finger making a gesture is moved, changing the properties of the gesture.
gestureend This high-level event is fired as soon as the gesture ends, for example because one of the two fingers touching the touch screen are lifted.
gesturestart This high-level event is fired when two or more fingers touch the touch screen and are used to make a "gesture".
longpress This event is fired after a finger has been pressed against the screen for a certain amount of time without starting a drag or gesture.
tap This event is fired after a touchstart and touchend has occurred at the same target and only if neither of the touchstart nor the touchend event has been canceled and no gesture was started in between.
touchend This event is fired when a finger is lifted from a touch screen.
touchmove This event is fired when a finger touches and moves on a touch screen.
touchstart This event is fired when a finger is touches a touch screen.

Interface Description

Each class implementing this interface declares that it can act as the target for certain events. Each event in this interface (and therefore also within the classes implementing the interface) represents an event of a specific type.

The following example shows event handling for events of the type "click":

here

```
// Note that "obj" can be either a DOM node or any other JavaScript object.
var obj = nokia.maps.dom.EventTarget( {} );
obj.addListener("click", function (evt) {
  console.log("This is the '"+evt.type+"' event!");
});

obj.dispatch( new nokia.maps.dom.Event({
  type: "click"
}));
```

The example creates an instant of `nokia.maps.dom.EventTarget`, adds a listener for "click" events to it, and dispatches a "click" event to all registered listeners (in this case one) - the listeners receive the object that represents the target of the event.

For more information about dispatching events, please refer to the documentation of the [nokia.maps.dom.EventTarget](#).

Event Details

dblTap

This event is fired after a two tap events have been fired in a certain amount of time.

Event Handler Parameters:

evt	{ nokia.maps.dom.TouchEvent }
	An object representing the event

gesturechange

This high-level event is fired whenever either of the two finger making a gesture is moved, changing the properties of the gesture.

Event Handler Parameters:

evt	{ nokia.maps.dom.TouchEvent }
	An object representing the event

gestureend

This high-level event is fired as soon as the gesture ends, for example because one of the two fingers touching the touch screen are lifted.

here

Event Handler Parameters:

evt {*nokia.maps.dom.TouchEvent*}

An object representing the event

gesturestart

This high-level event is fired when two or more fingers touch the touch screen and are used to make a "gesture".

Event Handler Parameters:

evt {*nokia.maps.dom.TouchEvent*}

An object representing the event

longpress

This event is fired after a finger has been pressed against the screen for a certain amount of time without starting a drag or gesture.

Event Handler Parameters:

evt {*nokia.maps.dom.TouchEvent*}

An object representing the event

tap

This event is fired after a touchstart and touchend has occurred at the same target and only if neither of the touchstart nor the touchend event has been canceled and no gesture was started in between.

Event Handler Parameters:

evt {*nokia.maps.dom.TouchEvent*}

An object representing the event

here

touchend

This event is fired when a finger is lifted from a touch screen.

Event Handler Parameters:

evt {*nokia.maps.dom.TouchEvent*}

An object representing the event

touchmove

This event is fired when a finger touches and moves on a touch screen. This event is different from a mousemove event, because the user can touch the screen with more than one finger at the same time.

Event Handler Parameters:

evt {*nokia.maps.dom.TouchEvent*}

An object representing the event

touchstart

This event is fired when a finger touches a touch screen.

Event Handler Parameters:

evt {*nokia.maps.dom.TouchEvent*}

An object representing the event

Interface: WheelEvent

This interface is a member of *nokia.maps.dom*.

Extends: *nokia.maps.dom.Event*

here

Interface Summary

This event is fired when mouse wheel is used.

[For full details, see [nokia.maps.dom.WheelEvent](#)]

Table 100: Property Summary

Properties
altKey: {Boolean} This field indicates the activation state of the modifier key Alt.
clientX: {Number} This property holds the horizontal coordinate at which the event occurred relative to the viewport associated with the event, if the wheel is associated with a pointing device.
clientY: {Number} This property holds the vertical coordinate at which the event occurred relative to the viewport associated with the event, if the wheel is associated with a pointing device.
ctrlKey: {Boolean} This field indicates the activation state of the modifier key Ctrl.
readOnly deltaMode: {Number} This property indicates the units of measurement for the delta values (rotation distance).
readOnly deltaX: {Number} This property indicates the distance the wheel has rotated around the x-axis for a mouse-/wheel event or the number of pixels the mouse was moved since the last drag or dragover event within a drag operation.
readOnly deltaY: {Number} This property indicates the distance the wheel has rotated around the y-axis for a mouse-/wheel event or the amount of pixel the mouse was moved since the last drag or dragover event within a drag operation.
readOnly deltaZ: {Number} This property indicates the distance the wheel has rotated around the z-axis.
readOnly DOM_DELTA_LINE: {Number} This field is an identifier specifying the units of measurement for the delta (rotation distance) as individual lines of text.
readOnly DOM_DELTA_PAGE: {Number} This field is an identifier specifying the units of measurement for the delta (rotation distance) as pages, either defined as a single screen or as a demarcated page.
readOnly DOM_DELTA_PIXEL: {Number} This field is an identifier specifying the units of measurement for the delta (rotation distance) as pixels.
metaKey: {Boolean} This field indicates the activation state of the modifier key Meta.

here

Properties

pageX: {Number}

This property holds the horizontal coordinate at which the event occurred relative to the document associated with the event, if the wheel is associated with a pointing device.

pageY: {Number}

This property holds the vertical coordinate at which the event occurred relative to the document associated with the event, if the wheel is associated with a pointing device.

screenX: {Number}

This property holds the horizontal coordinate at which the event occurred relative to the origin of the screen coordinate system, if the wheel is associated with a pointing device.

screenY: {Number}

This property holds the vertical coordinate at which the event occurred relative to the origin of the screen coordinate system, if the wheel is associated with a pointing device.

shiftKey: {Boolean}

This field indicates the activation state of the modifier key Shift.

targetX: {Number}

This property holds the x-position of the cursor relative to the target, if the wheel is associated with a pointing device.

targetY: {Number}

This property holds the y-position of the cursor relative to the target, if the wheel is associated with a pointing device.

Directly Inherited Properties

Inherited from class [nokia.maps.dom.Event](#) :

[AT_TARGET](#), [bubbles](#), [BUBBLING_PHASE](#), [canBubble](#), [cancelable](#), [canSicker](#), [CAPTURING_PHASE](#), [currentTarget](#), [defaultPrevented](#), [eventPhase](#), [namespaceURI](#), [nativeEvent](#), [page](#), [propagation](#), [PROPAGATION_OK](#), [PROPAGATION_STOP](#), [PROPAGATION_STOP_IMMEDIATE](#), [target](#), [timeStamp](#), [type](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.dom.Event](#) :

[cancel](#), [clone](#), [preventDefault](#), [preventUnload](#), [stopImmediatePropagation](#), [stopPropagation](#)

here

Interface Description

This event indicates that a wheel on a pointing device has been rotated. It typically reflects the fact that the user has scrolled the mouse wheel, but it may also refer to scrollable elements of other pointing devices, for example trackballs.

Note that this class represents an unimplemented interface. It is included here to document its properties. The class is derived from [nokia.maps.dom.Event](#), but, in addition to the properties of the parent class, it has the properties described here.

Property Details

altKey: {Boolean}

This field indicates the activation state of the modifier key Alt. It is set to `true` if the modifier is activated.

clientX: {Number}

This property holds the horizontal coordinate at which the event occurred relative to the viewport associated with the event, if the wheel is associated with a pointing device.

clientY: {Number}

This property holds the vertical coordinate at which the event occurred relative to the viewport associated with the event, if the wheel is associated with a pointing device.

ctrlKey: {Boolean}

This field indicates the activation state of the modifier key Ctrl. It is set to `true` if the modifier is activated.

readonly **deltaMode:** {Number}

This property indicates the units of measurement for the delta values (rotation distance). The default value is `DOM_DELTA_PIXEL` (pixels). The value of this property may be different for each of `deltaX`, `deltaY`, and `deltaZ`, based on system configuration.

readonly **deltaX:** {Number}

here

This property indicates the distance the wheel has rotated around the x-axis for a mouse-/wheel event or the number of pixels the mouse was moved since the last drag or dragover event within a drag operation.

readonly **deltaY**: {Number}

This property indicates the distance the wheel has rotated around the y-axis for a mouse-/wheel event or the amount of pixel the mouse was moved since the last drag or dragover event within a drag operation.

readonly **deltaZ**: {Number}

This property indicates the distance the wheel has rotated around the z-axis.

readonly **DOM_DELTA_LINE**: {Number}

This field is an identifier specifying the units of measurement for the delta (rotation distance) as individual lines of text. This is the case for many form controls.

readonly **DOM_DELTA_PAGE**: {Number}

This field is an identifier specifying the units of measurement for the delta (rotation distance) as pages, either defined as a single screen or as a demarcated page.

readonly **DOM_DELTA_PIXEL**: {Number}

This field is an identifier specifying the units of measurement for the delta (rotation distance) as pixels. This is the most typical case in most operating system and implementation configurations.

metaKey: {Boolean}

This field indicates the activation state of the modifier key Meta. It is set to true if the modifier is activated.

pageX: {Number}

This property holds the horizontal coordinate at which the event occurred relative to the document associated with the event, if the wheel is associated with a pointing device.

here

This is a proprietary member and not part of the W3C interface specification this class implements.

pageY: {Number}

This property holds the vertical coordinate at which the event occurred relative to the document associated with the event, if the wheel is associated with a pointing device.

This is a proprietary member and not part of the W3C interface specification this class implements.

screenX: {Number}

This property holds the horizontal coordinate at which the event occurred relative to the origin of the screen coordinate system, if the wheel is associated with a pointing device.

screenY: {Number}

This property holds the vertical coordinate at which the event occurred relative to the origin of the screen coordinate system, if the wheel is associated with a pointing device.

shiftKey: {Boolean}

This field indicates the activation state of the modifier key Shift. It is set to `true` if the modifier is activated.

targetX: {Number}

This property holds the x-position of the cursor relative to the target, if the wheel is associated with a pointing device.

This is a proprietary member and not part of the W3C interface specification this class implements.

targetY: {Number}

This property holds the y-position of the cursor relative to the target, if the wheel is associated with a pointing device.

This is a proprietary member and not part of the W3C interface specification this class implements.

here

Namespace: geo

This namespace is a member of [nokia.maps](#).

Namespace Summary

This namespace offers facilities that allow you to create and manage objects and entities within the geographic coordinate system.

Namespace Description

This namespace offers facilities that allow you to create and manage objects and entities within the geographic coordinate system.

Class: BoundingBox

This class is a member of [nokia.maps.geo](#).

Class Summary

This class represents a rectangular area specified by two geographic coordinates.

[For full details, see [nokia.maps.geo.BoundingBox](#)]

Table 101: Property Summary

Properties
readonly bottomRight : { nokia.maps.geo.Coordinate } This property holds an object containing geographic coordinates of the bottom right of the bounding box.
readonly isCDB : {Boolean} This property holds a flag indicating whether the bounding box crosses the date border (true) or not (false).
readonly topLeft : { nokia.maps.geo.Coordinate } This property holds an object containing geographic coordinates of the top left of the bounding box.

Table 102: Method Summary

Methods
contains (bbox) : {Boolean} This method checks if the object supplied by the caller lies within the area of the given bounding box.
static coverAll (coordinates) : { nokia.maps.geo.BoundingBox }

here

Methods

This constructs a bounding box from an array of point objects (instances of `Coordinate`).

static **fromObject** (obj, skipValidation) : {*nokia.maps.geo.BoundingBox*}

This method constructs an instance of `BoundingBox` from a set of parameters supplied by the caller.

static **fromPath** (path, skipValidation) : {*nokia.maps.geo.BoundingBox*}

This method constructs a bounding box from a `Strip` object that represents a path.

getCenter () : {*nokia.maps.geo.Coordinate*}

This method returns an object containing the coordinates of the center of the given bounding box.

getHeight () : {Number}

This method returns the height of the bounding box in decimal degrees.

getWidth () : {Number}

This method returns the width of the bounding box in decimal degrees.

intersects (bbox)

This method checks if the intersection of two bounding boxes is non-empty.

isEmpty ()

The method checks if the area enclosed by the given bounding box is 0.

static **merge** (boxes) : {*nokia.maps.geo.BoundingBox*}

This method returns the smallest bounding box that covers all given boxes.

merge (boxes)

This method returns the smallest bounding box that covers the given bounding box and those supplied by the caller.

resizeToCenter (center) : {*nokia.maps.geo.BoundingBox*}

This method clones the given bounding box and resizes the clone if necessary until the location supplied by the caller is at its center.

Class Description

This class represents a rectangular area defined in terms of the geographic coordinates its top-left and bottom-right corners. A bounding box is not necessarily the smallest rectangle spanned by the two points. A bounding box wider than 180° or higher than 90° can be defined by setting the longitude of the top-left corner to a larger value than the longitude of the bottom-right corner. An instance of `BoundingBox` is immutable.

Constructor Details

nokia.maps.geo.BoundingBox(topLeft, bottomRight, skipValidation)

This method initializes a new instance of `BoundingBox`.

here

Parameters:

- topLeft:** `{nokia.maps.geo.Coordinate}`
- An object containing the geographical coordinates of the top left corner; latitude must be greater or at least the same as the latitude of `bottomRight`; longitude should be smaller than the longitude of `bottomRight`
- bottomRight:** `{nokia.maps.geo.Coordinate}` [optional]
- An object containing the geographical coordinates of the bottom right corner; latitude must be lower or at least the same as the latitude of the top left corner; longitude should be greater than the longitude of the top left corner
- skipValidation:** `{Boolean}` [optional]
- A flag indicating if the validation of the latitude is to be omitted; `true` means that no validation occurs, in which case the caller must ensure that the latitude of the bottom right corner is greater than that of the top left

Property Details

readonly **bottomRight:** `{nokia.maps.geo.Coordinate}`

This property holds an object containing geographic coordinates of the bottom right of the bounding box.

readonly **isCDB:** `{Boolean}`

This property holds a flag indicating whether the bounding box crosses the date border (`true`) or not (`false`).

Note that a bounding box that spans the globe from -180° (West, on the left) to $+180^{\circ}$ (East, on the right) is not considered to cross the International Date Line (which roughly follows the 180° longitude).

Default Value: `false`

readonly **topLeft:** `{nokia.maps.geo.Coordinate}`

This property holds an object containing geographic coordinates of the top left of the bounding box.

here

Method Details

contains(bbox): {Boolean}

This method checks if the object supplied by the caller lies within the area of the given bounding box. The object to check may either represent a point on the map or a bounding box.

Parameters:

bbox: {*nokia.maps.geo.Coordinate* | *nokia.maps.geo.BoundingBox*}

An object representing a point on the map or a bounding box

Returns:

{Boolean} A boolean value, true indicates that the object supplied by the caller is contained within the given bounding box, while false indicates that the received object is not contained within the bounding box

static coverAll(coordinates): {*nokia.maps.geo.BoundingBox*}

This constructs a bounding box from an array of point objects (instances of *Coordinate*). If the method is successful, the caller receives the smallest bounding box that which contains all the points.

Parameters:

coordinates: {*nokia.maps.geo.ICoordinate*[]}

An array of point objects (instances of *Coordinate*)

Returns:

{*nokia.maps.geo.BoundingBox*}

The calculated bounding box

static fromObject(obj, skipValidation): {*nokia.maps.geo.BoundingBox*}

This method constructs an instance of *BoundingBox* from a set of parameters supplied by the caller. The method accepts one parameter, which must be an array with four values top, left, bottom, right (lat, lng, lat, lng), or an array with two elements, each an instance of *geo.Coordinate*. All latitude/longitude values must be given in decimal degrees (WGS84). The parameter can also be an instance of *BoundingBox*.

here

Parameters:

obj: {*nokia.maps.geo.BoundingBox* | Array}

An array containing two point objects (instances of *Coordinate*), or an array of four coordinates (lat, lng, lat, lng), or a bounding box

skipValidation: {Boolean} [optional, default: false]

If true, then validation of latitude values is omitted when creating the bounding box; see also the constructor

Returns:

{*nokia.maps.geo.BoundingBox*}

The new geo bounding box or null if no bounding box could be created from the given parameter

static **fromPath**(path, skipValidation): {*nokia.maps.geo.BoundingBox*}

This method constructs a bounding box from a *Strip* object that represents a path. The caller receives a bounding box object which contains the path.

Parameters:

path: {*nokia.maps.geo.Strip*}

A *Strip* from which to create a bounding box object

skipValidation: {Boolean} [optional]

A Boolean, if true, validation of latitude is omitted

Returns:

{*nokia.maps.geo.BoundingBox*}

The calculated geo bounding box

getCenter(): {*nokia.maps.geo.Coordinate*}

This method returns an object containing the coordinates of the center of the given bounding box.

here

Returns:

`{nokia.maps.geo.Coordinate}`

An instance of `Coordinate` representing the center of the bounding box on which the method has been called

getHeight(): `{Number}`

This method returns the height of the bounding box in decimal degrees.

Returns:

`{Number}` The height of the bounding box in decimal degrees.

getWidth(): `{Number}`

This method returns the width of the bounding box in decimal degrees.

Returns:

`{Number}` The width of the bounding box in decimal degrees.

intersects(bbox)

This method checks if the intersection of two bounding boxes is non-empty.

Parameters:

bbox: `{nokia.maps.geo.BoundingBox}`

A `BoundingBox` object to be tested for intersection with the bounding box on which the method is called

Returns:

A `{Boolean}` indicating if the two bounding boxes intersect (`true`) or not (`false`)

isEmpty()

The method checks if the area enclosed by the given bounding box is 0.

here

Returns:

A {Boolean} indicating if the dimension of the bounding box is 0 (true) or not (false)

static **merge**(boxes): {*nokia.maps.geo.BoundingBox*}

This method returns the smallest bounding box that covers all given boxes.

Parameters:

boxes: {*nokia.maps.geo.BoundingBox*[]}

An array of BoundingBox instances to include in the new bounding box

Returns:

{*nokia.maps.geo.BoundingBox*}

A new instance of BoundingBox that contains the array of bounding boxes supplied by the caller

merge(boxes)

This method returns the smallest bounding box that covers the given bounding box and those supplied by the caller.

Parameters:

boxes: {*nokia.maps.geo.BoundingBox* | *nokia.maps.geo.BoundingBox*[]}

An array of instance of BoundingBox

Returns:

An instance of {*nokia.maps.geo.BoundingBox*} representing the smallest rectangular area that includes the bounding box on which the method was called as well as the bounding boxes provided by the caller

resizeToCenter(center): {*nokia.maps.geo.BoundingBox*}

here

This method clones the given bounding box and resizes the clone if necessary until the location supplied by the caller is at its center.

Parameters:

center: {[nokia.maps.geo.Coordinate](#)}

A point which is to be the center of the resized bounding box.

Returns:

{[nokia.maps.geo.BoundingBox](#)}

The resized bounding box

Class: Coordinate

This class is a member of [nokia.maps.geo](#).

Class Summary

This class represents a geographical location in WGS84 coordinate system.

[For full details, see [nokia.maps.geo.Coordinate](#)]

Table 103: Property Summary

Properties
static altModes : {String[]}
Array of possible altitude modes defined in nokia.maps.geo.ICoordinate#altMode

Directly Inherited Properties

Inherited from class [nokia.maps.geo.ICoordinate](#) :

[altitude](#), [altMode](#), [latitude](#), [longitude](#)

Table 104: Method Summary

Methods
distance (coord, straight) : {Number}
This method calculates the distance between the location represented by the given instance of Coordinate and that supplied by the caller.
equals (other) : {Boolean}

here

Methods

This method checks if the location supplied by the caller is the same as the location represented by the given instance of `Coordinate`.

static `fromObject ()`

This method attempts to create an instance of `Coordinate` from the supplied arguments.

static `isValid (lat, lng, alt, altMode) : {Boolean}`

This method checks if the received arguments are numbers and in valid ranges.

`toString () : {String}`

This method creates a string representation of the given instance of `Coordinate`.

`walk (bearing, distance, overGreatCircle) : {nokia.maps.geo.Coordinate}`

This method calculates the geographic coordinates of a destination point using the distance and bearing specified by the caller.

Class Description

This class represents a geographical location, a point on the map, defined in terms of its geographical coordinates latitude, longitude and (optionally) altitude. If a corresponding type is already provided by the platform, the product must not implement this class.

Constructor Details

`nokia.maps.geo.Coordinate(lat, lng, alt, skipValidation, altMode)`

This method initializes a new instance of `Coordinate`

Parameters:

lat: {Number}

Latitude in degrees; values must be in the range [-90, 90], otherwise the error `IllegalArgumentException` is thrown

lng: {Number}

Longitude in degrees; values must be in the range [-180, 180], otherwise the error `IllegalArgumentException` is thrown

alt: {Number} [optional, default: undefined]

Altitude in meters

skipValidation: {Boolean} [optional, default: false]

here

If true, validation of the latitude, longitude and altitude is not performed

altMode: {String} [optional]

Altitude mode as string. If omitted "relative to ground level" is used. See [nokia.maps.geo.ICoordinate#altMode](#) for valid values.

Property Details

static altModes: {String[]}

Array of possible altitude modes defined in [nokia.maps.geo.ICoordinate#altMode](#)

Method Details

distance(coord, straight): {Number}

This method calculates the distance between the location represented by the given instance of `Coordinate` and that supplied by the caller. The method uses the Haversine formula. Altitude is not considered.

Parameters:

coord: {[nokia.maps.geo.ICoordinate](#)}

An object representing the location to which the distance is to be calculated

straight: {Boolean} [optional]

An flag to compute distance in 3D space, not shortest distance on sphere.

Returns:

{Number} The distance between the given location and the location supplied by the caller in meters

equals(other): {Boolean}

This method checks if the location supplied by the caller is the same as the location represented by the given instance of `Coordinate`. Only latitude and longitude are compared.

Parameters:

here

other: `{nokia.maps.geo.Coordinate}`

The location whose coordinates are to be compared against those in the given location object

Returns:

{Boolean} true if equal, otherwise false

static **fromObject()**

This method attempts to create an instance of `Coordinate` from the supplied arguments. Valid arguments are:

- an instance of `nokia.maps.geo.Coordinate` - if provided, no other arguments should be present
- two number values representing latitude and longitude (in this order)
- three number values representing latitude, longitude and altitude (in this order)
- an array of two numbers, where the first element is latitude and the second longitude
- an array of three numbers, where the first element is latitude, the second longitude and the third altitude
- an object containing the properties `latitude` and `longitude` - for example, `{latitude: 50.1299, longitude: 8.568}`
- an object containing the properties `lat` and `lng` - for example, `{lat: 50.1299, lng: 8.568}`

Returns:

A newly created instance of `{nokia.maps.geo.Coordinate}` or `null`

static **isValid(lat, lng, alt, altMode):** {Boolean}

This method checks if the received arguments are numbers and in valid ranges. Altitude should not be passed. If it is passed it must be a number not Infinity. Strings are not supported.

Parameters:

lat: {Number}

The latitude in decimal degrees to validate

here

lng: {Number}
The longitude in decimal degrees to validate

alt: {Number} [optional]
If supplied, the altitude in meters to validate

altMode: {String} [optional]
If supplied, the altitude mode to validate

Returns:
{Boolean} true if the given values are valid; false otherwise

toString(): {String}

This method creates a string representation of the given instance of `Coordinate`.

Returns:
{String} The resulting string representation of the given location point object

walk(bearing, distance, overGreatCircle): {[nokia.maps.geo.Coordinate](#)}

This method calculates the geographic coordinates of a destination point using the distance and bearing specified by the caller.

Parameters:

bearing: {Number}
The bearing to use in the calculation in degrees

distance: {Number}
The distance to the destination in meters

overGreatCircle: {Boolean} [optional, default: false]
If true the computation uses the "Great Circle" otherwise "Rhumb Line".

Returns:

here

`{nokia.maps.geo.Coordinate}`

The calculated target as a location object containing geo coordinates

Interface: Corridor

This interface is a member of [nokia.maps.geo](#).

Interface Summary

This interface represents a belt that follows the path sweeping equal areas either side of this path.

[For full details, see [nokia.maps.geo.Corridor](#)]

Table 105: Property Summary

Properties
line: {nokia.maps.geo.Shape} This property holds a list of point objects (each defined in terms of its latitude and longitude).
radius: {Number} This property specifies the width of the corridor in meters.

Interface Description

This interface represents a belt that follows the path of a polyline, sweeping equal areas either side of the polyline. In routing, a corridor defines an area within which additional navigation can be provided to help the driver/pedestrian return to the calculated route after one or more wrong turns.

Property Details

line: {[nokia.maps.geo.Shape](#)}

This property holds a list of point objects (each defined in terms of its latitude and longitude). The polyline connecting those points marks the path of the corridor.

radius: {Number}

This property specifies the width of the corridor in meters.

Interface: ICoordinate

This interface is a member of [nokia.maps.geo](#).

here

Interface Summary

This interface defines the properties of an instance of `Coordinate`.

[For full details, see [nokia.maps.geo.ICoordinate](#)]

Table 106: Property Summary

Properties
readonly altitude : {Number undefined} This property holds the value of altitude in meters.
readonly altMode : {String undefined} This property holds a string indicating the altitude mode for the given instance of <code>ICoordinate</code> .
readonly latitude : {Number} This property holds the value of latitude as a numeric value in the range from -90 (south) to +90 (north) degrees.
readonly longitude : {Number} This property holds the value of longitude as a numeric value in the range from -180 (west) to +180 (east) degrees.

Interface Description

This interface defines the properties of an instance of `Coordinate`.

Property Details

readonly **altitude**: {Number | undefined}

This property holds the value of altitude in meters. An undefined altitude is treated as 0.

readonly **altMode**: {String | undefined}

This property holds a string indicating the altitude mode for the given instance of `ICoordinate`.

Note that if the property is not set, it is assumed that altitude is relative to ground level. When set, the value of the property must be one of:

- "GL" - indicates that altitude is relative to ground level
- "OL" - indicates that altitude is relative to obstruction level
- "SL" - indicates that altitude is relative to mean sea level
- "SB" - indicates that altitude is relative to sea bed
- "WE" - indicates that altitude is relative to WGS84 ellipsoid
- "WG" - indicates that altitude is relative to WGS84 geoid

here

readonly **latitude**: {Number}

This property holds the value of latitude as a numeric value in the range from -90 (south) to +90 (north) degrees.

readonly **longitude**: {Number}

This property holds the value of longitude as a numeric value in the range from -180 (west) to +180 (east) degrees.

Interface: Proximity

This interface is a member of [nokia.maps.geo](#).

Interface Summary

This interface defines an area of proximity as a geographic area in the shape of a circle.

[For full details, see [nokia.maps.geo.Proximity](#)]

Table 107: Property Summary

Properties
center: {nokia.maps.geo.Coordinate} This property holds a mandatory reference to an instance of nokia.maps.geo.Coordinate that marks the center of the area of proximity.
radius: {Number} This property holds the radius in meters.

Interface Description

This interface defines an area of proximity as a geographic area in the shape of a circle.

Property Details

center: {[nokia.maps.geo.Coordinate](#)}

This property holds a mandatory reference to an instance of [nokia.maps.geo.Coordinate](#) that marks the center of the area of proximity.

here

radius: {Number}

This property holds the radius in meters.

Class: Shape

This class is a member of [nokia.maps.geo](#).

Extends: [nokia.maps.geo.Strip](#)

Class Summary

This class represents a shape defined in terms of the geographic coordinates of its vertices (a geographic shape).

[For full details, see [nokia.maps.geo.Shape](#)]

Table 108: Method Summary

Methods
destroy () This method clean up itself and stopping all asynchronous code.
static fromLatLngArray (arr, closed, callback, ctx) : { nokia.maps.geo.Shape } This method creates a shape from an array containing latitude and longitude (in decimal degrees) for all the points that are to define the shape.

Directly Inherited Methods

Inherited from class [nokia.maps.geo.Strip](#) :

[add](#), [addAll](#), [addAllAsync](#), [addObserver](#), [asArray](#), [get](#), [getLatLng](#), [getLength](#), [remove](#), [removeObserver](#), [set](#), [splice](#)

Class Description

This class represents an observable geographic shape (derived from [nokia.maps.geo.Strip](#)) with a geometry contained in a geographic quadtree and allows testing for containment, square-based clipping and near-by analysis.

Constructor Details

[nokia.maps.geo.Shape](#)(closed, depth)

This method initializes a new instance of Shape.

here

Parameters:

closed:	{Boolean} [optional, default: false]
	A Boolean that indicates if the shape is a polygon (true) or a polyline (false).
depth:	{Number} [optional, default: 10]
	A value that specifies the depth of the underlying quad tree.

Method Details

destroy()

This method clean up itself and stopping all asynchronous code. It is allways better to call this method when instance is not needed instead of throwing away instance reference only.

static **fromLatLngArray**(arr, closed, callback, ctx): {[nokia.maps.geo.Shape](#)}

This method creates a shape from an array containing latitude and longitude (in decimal degrees) for all the points that are to define the shape. The method executes synchronously or asynchronously, depending on whether the caller provides a callback function as one of the arguments.

Parameters:

arr:	{Number []}
	An array with latitude/longitude coordinates in decimal degrees
closed:	{Boolean}
	A flag indicating if the shape is closed (true) or not (false)
callback:	{Function} [optional]
	A caller-defined callback function; if provided, the method runs asynchronously and calls the callback as soon as it is finished; this is done for large coordinate arrays; The callback receives the generated shape instance and the supplied context as parameters
ctx:	{Object} [optional]
	The context in which the callback function is to be called; if not provided, the callback is called in the context of the window

here

Returns:

`{nokia.maps.geo.Shape}`

If the method is called synchronously (without a callback function) it returns the generated shape, otherwise it will return `null`

Class: Strip

This class is a member of [nokia.maps.geo](#).

Class Summary

This class represents an array of locations defined by their geographic coordinates.

[For full details, see [nokia.maps.geo.Strip](#)]

Table 109: Method Summary

Methods
add (coord, idx) This method adds a single element to the given <code>Strip</code> object.
addAll (coords, idx, mode) This method adds a collection of coordinates to the given <code>Strip</code> instance.
addAllAsync (coords, idx, mode, callback) This method adds a collection of coordinates to the given <code>Strip</code> instance asynchronously.
addObserver (callback, context) This method registers a new <code>Strip</code> observer.
asArray () : {Array[Number[]]} This method retrieves all elements of the given <code>strip</code> object as an array.
static convertToArray (coords, mode) : {Array[Number[]]} This method converts an array of objects or numbers to an array containing latitude, longitude and altitude, depending on the specified conversion mode.
static fromObject (value) : { nokia.maps.geo.Strip } This method creates a new instance of <code>Strip</code> on the basis of the received value.
get (idx) : { nokia.maps.geo.Coordinate } This method retrieves a <code>strip</code> element at the specified index.
getLatLng (idx, count) : {Number[]} This method retrieves latitudes and longitudes from <code>strip</code> elements, starting at the caller-specified index.

here

Methods

`getLength` () : {Number}

This method retrieves the number of elements held in the given strip object.

`remove` (idx)

This method removes a single element at a specified index in the strip.

`removeObserver` (callback, context) : {*nokia.maps.util.Object*}

This method unregisters a Strip observer.

`set` (idx, coord)

This method inserts an object containing the geographic coordinates of a location at the specified index in the given strip object.

`splice` (idx, remove, coords, mode)

This method splices an array of coordinates into the given Strip instance or removes elements from the Strip.

Class Description

This class represents an array of locations defined by their geographic coordinates. The array optimizes the storage of the data it holds and supports specialized operations on them.

Constructor Details

`nokia.maps.geo.Strip`(coords, mode)

This method initializes a new instance of Strip.

Parameters:

coords: {Number[] | *nokia.maps.geo.Coordinate*[]}

Either an array of geographic coordinates following the pattern [lat, lng, alt, ..., lat, lng, alt] or an array of instances of {@link *nokia.maps.geo.Coordinate*}

mode: {String}

A string indicating how the array is to be interpreted, for details, please see {@link *nokia.maps.geo.Strip.convertToArray*}().

Method Details

`add`(coord, idx)

This method adds a single element to the given Strip object.

here

Parameters:

coord:	<code>{nokia.maps.geo.Coordinate}</code>
	An object representing an element to be added to the <code>Strip</code>
idx:	<code>{Number}</code> [optional]
	An optional index at which the element is to be inserted

addAll(coords, idx, mode)

This method adds a collection of coordinates to the given `Strip` instance.

Parameters:

coords:	<code>{Array}</code>
	A collection containing coordinates to be added to the <code>Strip</code> ; it can be either an array of latitude and longitude values or an array of objects that can be converted to instances of nokia.maps.geo.Coordinate .
idx:	<code>{Number}</code> [optional]
	The optional index at which the received collection of coordinates is to be inserted
mode:	<code>{String}</code> [optional]
	The optional mode indicating how coords should be interpreted (for details, please see nokia.maps.geo.Strip.convertToArray())

addAllAsync(coords, idx, mode, callback)

This method adds a collection of coordinates to the given `Strip` instance asynchronously.

Parameters:

coords:	<code>{Array}</code>
	The collection of coordinates to be added to the <code>Strip</code> ; it can be either an array of latitude and longitude values or an array of objects that can be converted to instances of nokia.maps.geo.Coordinate

here

idx:	{Number} [optional]
	The optional index at which the collection is to be inserted
mode:	{String} [optional]
	The optional mode indicating how coords should be interpreted (for details please see nokia.maps.geo.Strip.convertToArray())
callback:	{Function} [optional]
	The optional callback function to be called when the operation is complete

addObserver(callback, context)

This method registers a new Strip observer.

Parameters:

callback:	{Function}
	A callback function to be invoked (when the Strip has been modified). The function must be able to accept the following arguments: • (Strip) strip - the Strip instance itself • (String) idx - the start index of the changes • (Number) inserted - a value indicating how many elements that have been inserted • (Array[Number]) removed - the coordinates that have been removed as an array of the form [lat, lng, alt, ..., lat, lng, alt]
context:	{Object} [optional, default: null]
	The context to use when the callback is invoked.

asArray(): {Array[Number]}

This method retrieves all elements of the given strip object as an array. The array format is [lat, lng, alt, ..., lat, lng, alt].

Returns:

here

`{Array[Number]}` An array of coordinate values (a copy of the given strip object's internal array).

`static convertToArray(coords, mode): {Array[Number]}`

This method converts an array of objects or numbers to an array containing latitude, longitude and altitude, depending on the specified conversion mode. The array has the form `[lat, lng, alt, ..., lat, lng, alt]`.

Parameters:

`coords:` `{Array}` [optional]

An optional collection of geographic coordinates to be inserted into the `Strip` after the start index; this argument can be either an array of latitude and longitude values or an array of objects that can be converted to instances of [nokia.maps.geo.Coordinate](#)

`mode:` `{String}` [optional]

An optional string indicating how the received array should be interpreted. The default value is "auto". The possible values are:

- "auto" - if the array consists of numbers, it is interpreted as an array of the form `[lat, lng, ..., lat, lng]`; if the caller passes in an instance of `Strip`, the internal array of geographic points held in the received object is used; otherwise, the method attempts to convert each element of the received array to instances of `{@link nokia.maps.geo.Coordinate}`
- "coords" - the method attempts to convert each element of the received array to instances of [nokia.maps.geo.Coordinate](#), using the method [nokia.maps.geo.Coordinate\(\)](#)
- "values lat lng alt" - the method interprets the received array as a array of values, where latitude, longitude and altitude are given in the order in which "lat", "lng" and "alt" appear in the mode string; "alt" is optional in the mode string; for example, "values lat lng alt" is interpreted to mean that the array has the format `[lat, lng, alt, ..., lat, lng, alt]`

here

- "arrays lat lng alt" - the method interprets the received array as containing sub-arrays, one with latitude values, one with longitude values and one with altitude values; these sub-arrays are ordered to mirror the order in which "lat", "lng" and "alt" appear in the mode string; "alt" is optional; for example, "arrays lat lng alt" is interpreted to mean that the array has the format `[[lat, lat, ...], [lng, lng, ...], [alt, alt, ...]]`

Additionally, the first element in the mode string can the keyword "unsafe" to prevent the method from running time-consuming input validation. This option should be used when the value of the argument `coords` comes from a trusted source. When "unsafe" precedes "auto" or "coords", [`nokia.maps.geo.Coordinate.fromObject\(\)`](#) is not called, therefore no validation is performed and the received objects are assumed to contain the properties `latitude`, `longitude` and `altitude` (as in `{@link nokia.maps.geo.Coordinate}`). The fastest mode to use is "unsafe values lat lng alt".

Returns:

`{Array[Number]}` An array in the form `[lat, lng, alt, ..., lat, lng, alt]`.

`static fromObject(value): {nokia.maps.geo.Strip}`

This method creates a new instance of `Strip` on the basis of the received value. If the caller provides an existing instance `Strip`, the same object is returned. If the caller provides an array, the array is converted to a `Strip` (note that mode must be the first item in the array -- please see [`nokia.maps.geo.Strip.convertToArray\(\)`](#) for details). When it receives any other type of value/object, the method returns an empty `Strip`.

Parameters:

value: `{Object}`

An object or a value from which to create a `Strip` instance.

Returns:

`{nokia.maps.geo.Strip}`

here

An instance of `Strip` created on the basis of the object supplied by the caller.

get(idx): {*nokia.maps.geo.Coordinate*}

This method retrieves a strip element at the specified index.

Parameters:

idx: {Number}

The index from which the element should be retrieved.

Returns:

{*nokia.maps.geo.Coordinate*}

An object containing the geographic coordinates of a location.

getLatLng(idx, count): {Number[]}

This method retrieves latitudes and longitudes from strip elements, starting at the caller-specified index.

Parameters:

idx: {Number}

The index from which the coordinates should be retrieved.

count: {Number} [optional, default: 1]

The number of elements whose coordinates are to be retrieved; if the value is negative, the returned coordinates are presented in reverse order.

Returns:

{Number[]} An array of coordinates in the form [lat, lng, lat, lng, ...].

getLength(): {Number}

This method retrieves the number of elements held in the given strip object.

here

Returns:

`{Number}` The length of the strip as a number of elements.

remove(idx)

This method removes a single element at a specified index in the strip.

Parameters:

idx: `{Number}`
 The index at which the element is to be removed

removeObserver(callback, context): `{nokia.maps.util.Object}`

This method unregisters a Strip observer.

Parameters:

callback: `{Function}`
 The observer callback function to be removed.

context: `{Object}`
 The context object for the callback.

Returns:

`{nokia.maps.util.Object}`
 A reference to the given instance of Strip after the callback has been removed (an instance of *Ob*ject)

set(idx, coord)

This method inserts an object containing the geographic coordinates of a location at the specified index in the given strip object. A pre-existing element at the specified index is replaced. To extend the strip, the insertion index must be equal to the index of the last element plus one - assuming a strip of n elements, the index must be n+1. An attempt use an index greater than n+1 results in an exception.

here

Parameters:

idx: {Number}

The index at which the element is to be inserted; if a strip element already exists at that index, it is replaced

coord: {[nokia.maps.geo.Coordinate](#)}

An object representing a location defined in terms of latitude and longitude to be added to the strip

splice(idx, remove, coords, mode)

This method splices an array of coordinates into the given `Strip` instance or removes elements from the `Strip`.

Parameters:

idx: {Number}

The start index after which elements should be removed and/or the collection is to be inserted

remove: {Number}

A value indicating how many elements should be removed after the start index

coords: {Array} [optional]

An optional collection of coordinates to be inserted into the `Strip`; it can be either an array of latitude and longitude values or an array of objects that can be converted to instances of `{@link nokia.maps.geo.Coordinate}`

mode: {String} [optional]

An optional string indicating how the received array (coords) should be interpreted; for details, please see [nokia.maps.geo.Strip.convertToArray\(\)](#) for further details.

here

Namespace: gfx

This namespace is a member of [nokia.maps](#).

Namespace Summary

This namespace defines classes and methods that provide facilities to create and manipulate graphics resources (images).

Namespace Description

This namespace defines classes and methods that provide facilities to create and manipulate graphics resources (images). The namespace provides support for bitmaps, SVGs and IDL (Image Description Language).

Class: BitmapImage

This class is a member of [nokia.maps.gfx](#).

Extends: [nokia.maps.gfx.Image](#)

Class Summary

This class loads a bitmap image from a Web server and makes it available as a GFX image.

[For full details, see [nokia.maps.gfx.BitmapImage](#)]

Table 110: Property Summary

Properties
readonly isBitmap : {Boolean} This bitfield property indicates that the given image is a bitmap.
readonly naturalHeight : {Number} This property holds the natural height of the image in pixels.
readonly naturalWidth : {Number} This property holds the natural width of the image in pixels.
readonly offsetX : {Number} This property holds the x-offset (in pixels) within the image to create a sprite or undefined, if the image is not to be used to create a sprite.
readonly offsetY : {Number}

here

Properties

This property holds the y-offset (in pixels) within the image to create a sprite or undefined, if the image is not to be used to create a sprite.

src: {String}

This property holds the URL of the image.

Directly Inherited Properties

Inherited from class [nokia.maps.gfx.Image](#) :

[height](#), [opacity](#), [state](#), [width](#)

Table 111: Method Summary

Methods

clone (doc, width, height, offsetX, offsetY) : {[nokia.maps.gfx.BitmapImage](#)}

This method clones the given image and clips it to the size specified by the caller (creating a sprite).

Directly Inherited Methods

Inherited from class [nokia.maps.gfx.Image](#) :

[clone](#), [createElement](#), [getDocument](#), [prepare](#), [setOpacity](#)

Class Description

The class `BitmapImage` loads a bitmap (PNG, GIF, JPEG and others) from a Web server and makes it available as a GFX image. The difference between a GFX image and a native DOM image is that a GFX image is not loaded unless it is necessary or forced by calling `prepare()`. Additionally, a GFX image can be clipped (sprite creation).

The main purpose of sprite images is to save server requests by adding multiple images into one image and then clip this single image to produce multiple images from one. GFX images are also used if it is unclear exactly when an image should be loaded. This applies for example to instances of [nokia.maps.map.Marker](#) and their icons. For example, you may add a thousand markers with different icon images to the map, but you only need to load those for the markers that are within the visible viewport.

```
// Create a reference to the document body.
var body = document.body || document._documentElement;

// Create a new bitmap.
var bitmap = new nokia.maps.gfx.BitmapImage("http://www.w3.org/Icons/WWW/w3c_home_nb.png");
```

here

```
// Add the bitmap in original size into the body.
body.appendChild(bitmap.createElement());

// Clone the bitmap, pick out the center of the image (from pixel 16,16 a
  square of 32 pixel)
// and add that to the document body.
var sprite = bitmap.clone(document, 32,32, 16,16);
body.appendChild(sprite.createElement());

// Display the size of the image as soon as it is known.
sprite.prepare(function (img) {
  alert("natural image size: " + img.naturalWidth + "x" + img.naturalHeight);
  alert("current sprite offset-x/y: " + img.offsetX + ", " + img.offsetY + "
    with " + img.width + "x" + img.height + "px size");
});
```

Constructor Details

nokia.maps.gfx.BitmapImage(image, doc, width, height, offsetX, offsetY)

This method creates a new image from an image URL or an already loaded [HTML image](#).

The method accepts four optional parameters for width, height, x- and y-offset to define a viewport for an image and to allow for spriting. If only the width and height are supplied, but no x- and y-offset, then the image is scaled to match these dimensions. If no width or height are provided by the caller at all, the image size remains unchanged.

Parameters:

image:	{String Image}
	Either a URL of an image or an already loaded HTML image
doc:	{Document} [optional]
	The document to which the instance of BitmapImage is to be bound; if this argument is omitted, the current document is used
width:	{Number} [optional]
	The target width of the image in pixels
height:	{Number} [optional]
	The target height of the image in pixels
offsetX:	{Number} [optional]
	The x-offset within the image to create a sprite (in pixels)
offsetY:	{Number} [optional]

here

The y-offset within the image to create a sprite (in pixels)

Property Details

readonly **isBitmap**: {Boolean}

This bitfield property indicates that the given image is a bitmap.

Default Value: true

readonly **naturalHeight**: {Number}

This property holds the natural height of the image in pixels. The value of the property may be undefined until the image has been prepared successfully.

readonly **naturalWidth**: {Number}

This property holds the natural width of the image in pixels. The value of the property may be undefined until the image has been prepared successfully.

readonly **offsetX**: {Number}

This property holds the x-offset (in pixels) within the image to create a sprite or undefined, if the image is not to be used to create a sprite.

readonly **offsetY**: {Number}

This property holds the y-offset (in pixels) within the image to create a sprite or undefined, if the image is not to be used to create a sprite.

src: {String}

This property holds the URL of the image.

Method Details

clone(doc, width, height, offsetX, offsetY): {*nokia.maps.gfx.BitmapImage*}

here

This method clones the given image and clips it to the size specified by the caller (creating a sprite). The clone is attached either to the same document as the original image, or to another document if provided by the caller.

Parameters:

doc:	{Document} [optional]	A reference to a document to which the image should be attached
width:	{Number} [optional]	The target width of the image in pixels; if not provided, the current value is used
height:	{Number} [optional]	The target height of the image in pixels; if not provided, the current value is used
offsetX:	{Number} [optional]	The x-offset within the image to create a sprite (in pixels); if not provided, the current value is used
offsetY:	{Number} [optional]	The y-offset within the image to create a sprite (in pixels); if not provided, the current value is used

Returns:

{[*nokia.maps.gfx.BitmapImage*](#)}

An object representing the cloned image

Class: CanvasPainter

This class is a member of [*nokia.maps.gfx*](#).

Extends: [*nokia.maps.gfx.Painter*](#)

Class Summary

This class render IDL into HTML5 Canvas tag.

here

[For full details, see [nokia.maps.gfx.CanvasPainter](#)]

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.gfx.Painter](#) :

[createElement](#), [setOpacity](#), [setScale](#), [updateElement](#)

Class Description

An instance of CanvasPainter renders a [nokia.maps.gfx.IDL](#) object into a DOM node using a CANVAS tag. Stable environments for the the canvas painter Nokia has tested include Firefox, Chrome and Safari (on Mac, PC and iOS 4.x). Recognized known issues in certain versions are, as a rule, fixed by the browser vendors at short notice.

To instantiate this class, use the default constructor without arguments:

```
var myCanvasPainter= new nokia.maps.gfx.CanvasPainter();
```

Constructor Details

nokia.maps.gfx.CanvasPainter()

This method creates a new instance of a CanvasPainter.

Namespace: Color

This namespace is a member of [nokia.maps.gfx](#).

Namespace Summary

The static Color class offers methods to handle 32-bit integer color values.

Table 112: Method Summary

Methods
static alpha (color) : {Number} This method retrieves the alpha byte of a compressed color.
static blue (color) : {Number} This method retrieves the blue byte of a compressed color.
static byteOf (percent) : {Number}

here

Methods

This method converts a percentage value (given as a decimal fraction between 0 and 1 into a byte value (0 to 255).

static **compress** (red, green, blue, alpha) : {Number}

This method compresses the red, green, blue and alpha bytes of a color into a 32-bit integer.

static **getCssHex** (color, addAlpha, prefix) : {String}

This method converts a compressed color into a hexadecimal CSS string notation, for example "#ff0000".

static **getCssRGB** (color, addAlpha) : {String}

This method converts a compressed color into a CSS rgb string, for example "rgb(255,0,0)".

static **getCssRGBA** (color) : {String}

This method converts a compressed color into a CSS rgba string, for example, "rgba(255,0,0,0.5)".

static **green** (color) : {Number}

This method retrieves the green byte of a compressed color.

static **parseCss** (cssColor, defaultOpacity) : {Number}

This method parses a CSS color definition into a compressed color (a 32-bit integer).

static **percentOf** (byteValue) : {Number}

This method converts a byte value (0 to 255) into a percentage value expressed as a decimal fraction between 0 and 1.

static **red** (color) : {Number}

This method retrieves the red byte of a compressed color.

Namespace Description

This class offers methods to handle 32-bit integer color values. A 32-bit integer offers the most efficient way to store a color and is used, for example, in [nokia.maps.gfx.IDL](#).

The methods on this class compress an RGBA (red, green, blue, alpha) color value into a 32-bit integer, extract values from a 32-bit integer, parse CSS strings into 32-bit integer color values or to create CSS strings from a 32-bit integer color value.

Method Details

static **alpha**(color): {Number}

This method retrieves the alpha byte of a compressed color.

Parameters:

color: {Number}

A color as a 32-bit integer

here

Returns:

{Number} The alpha part of the compressed color as a value between 0 and 255

static **blue**(color): {Number}

This method retrieves the blue byte of a compressed color.

Parameters:

color: {Number}
A color as a 32-bit integer

Returns:

{Number} The blue part of the compressed color as a value between 0 and 255

static **byteOf**(percent): {Number}

This method converts a percentage value (given as a decimal fraction between 0 and 1 into a byte value (0 to 255).

Parameters:

percent: {Number}
A decimal fraction between 0 and 1, representing a percentage value

Returns:

{Number} A number between 0 and 255, representing a byte value

static **compress**(red, green, blue, alpha): {Number}

This method compresses the red, green, blue and alpha bytes of a color into a 32-bit integer.

Parameters:

red: {Number}
The red byte of the color as a value between 0 and 255

here

green:	{Number}	The green byte of the color as a value between 0 and 255
blue:	{Number}	The blue byte of the color as a value between 0 and 255
alpha:	{Number}	The alpha byte of the color as a value between 0 and 255

Returns:

{Number}	The compressed 32-bit integer representing the color

Example:

```
// Compress a semi-transparent red color
var color = nokia.maps.gfx.Color.compress(255, 0, 0, 127);
```

static **getCssHex**(color, addAlpha, prefix): {String}

This method converts a compressed color into a hexadecimal CSS string notation, for example "#ff0000".

Parameters:

color:	{Number}	A 32-bit integer representing a compressed color value
addAlpha:	{Boolean} [optional]	A flag indicating whether to add the alpha byte (true) and thus an eight-character hex string instead or return a six-character RGB string
prefix:	{String} [optional]	The prefix to add in front of the returned hex string, the default is "#"

Returns:

{String}	A CSS hex string representing the given compressed color, for example "#ff0000"

here

```
static getCssRGB(color, addAlpha): {String}
```

This method converts a compressed color into a CSS rgb string, for example "rgb(255,0,0)".

Parameters:

color: {Number}

A 32-bit integer containing a compressed color value

addAlpha: {Boolean} [optional]

A flag indicating whether to return an rgba string (true) or an rgb string (false)

Returns:

{String} A valid CSS rgb color value, for example "rgb(255,0,0)"

```
static getCssRGBA(color): {String}
```

This method converts a compressed color into a CSS rgba string, for example, "rgba(255,0,0,0.5)".

Parameters:

color: {Number}

A 32-bit integer containing a compressed color value

Returns:

{String} A valid CSS rgba color value, for example "rgba(255,0,0,0.5)"

```
static green(color): {Number}
```

This method retrieves the green byte of a compressed color.

Parameters:

color: {Number}

A color as a 32-bit integer

here

Returns:

{Number} The green part of the compressed color as a value between 0 and 255

static **parseCss**(cssColor, defaultOpacity): {Number}

This method parses a CSS color definition into a compressed color (a 32-bit integer).

Parameters:

cssColor: {String}

The CSS color definition as a hexadecimal string, for example "#ff00ff".

defaultOpacity: {Number} [optional]

The default opacity of the color as a percentage (given as a decimal fraction between 0 and 1); the default is 1 (opaque); if no opacity is specified in the cssColor, then defaultOpacity is used

Returns:

{Number} The compressed color representation of the received hexadecimal string or null if the parsing failed

static **percentOf**(byteValue): {Number}

This method converts a byte value (0 to 255) into a percentage value expressed as a decimal fraction between 0 and 1.

Parameters:

byteValue: {Number}

A byte value between 0 and 255

Returns:

{Number} The corresponding percentage value as a double (a decimal fraction between 0 and 1)

static **red**(color): {Number}

here

This method retrieves the red byte of a compressed color.

Parameters:

color: {Number}
A color as a 32-bit integer

Returns:

{Number} The red part of the compressed color as a value between 0 and 255

Class: Graphics

This class is a member of *nokia.maps.gfx*.

Class Summary

This class represents a two-dimensional graphics context.

[For full details, see *nokia.maps.gfx.Graphics*]

Table 113: Method Summary

Methods
<i>beginImage</i> (width, height, description) : { <i>nokia.maps.gfx.Graphics</i> } This method resets the state of the graphics context and starts a new 2D image.
<i>beginPath</i> () : { <i>nokia.maps.gfx.Graphics</i> } This method clears all sub-paths and starts a new path.
<i>bezierCurveTo</i> (cp1x, cp1y, cp2x, cp2y, x, y) : { <i>nokia.maps.gfx.Graphics</i> } This method draws a Bezier curve from the current cursor position to a new position.
<i>closePath</i> () : { <i>nokia.maps.gfx.Graphics</i> } This method closes the current sub-path.
<i>drawEllipse</i> (x, y, w, h) : { <i>nokia.maps.gfx.Graphics</i> } This method creates a new ellipse path.
<i>drawRect</i> (x, y, w, h, rx, ry) : { <i>nokia.maps.gfx.Graphics</i> } This method creates a new rectangle sub-path and closes it.
<i>fill</i> () : { <i>nokia.maps.gfx.Graphics</i> } This method fills the current path.
<i>fillText</i> (text, x, y, nx, ny) : { <i>nokia.maps.gfx.Graphics</i> }

here

Methods

This method creates text path at the given position, oriented by a direction vector provided by the caller, closes it and then fills it with the fill color.

getDescription () : {String}

This method retrieves the description of the image.

getHeight () : {Number}

This method retrieves the height of the currently rendered image.

getIdl () : {*nokia.maps.gfx.IDL*}

This method retrieves the IDL of the given graphics context.

getWidth () : {Number}

This method retrieves the width of the currently rendered image.

lineTo (x, y) : {*nokia.maps.gfx.Graphics*}

This method draws a line from the current position to the pen position specified by the caller.

moveTo (x, y) : {*nokia.maps.gfx.Graphics*}

This method moves the cursor to the given position and starts a new sub-path.

polylineTo (coords) : {*nokia.maps.gfx.Graphics*}

This method draws a polyline from the current position along all the path specified by the caller as an array of locations.

quadraticCurveTo (cp_x, cp_y, x, y) : {*nokia.maps.gfx.Graphics*}

This method draws a quadratic curve from the current cursor position to a new position.

restore () : {*nokia.maps.gfx.Graphics*}

This method restores a previously saved state of the graphic context or, if the state has not been saved, replaces the current state with the default state.

rotate (angle) : {*nokia.maps.gfx.Graphics*}

This method rotates the current projection matrix (z axis rotation).

save () : {*nokia.maps.gfx.Graphics*}

This method saves the current state of the graphics context on the internal stack.

scale (sx, sy) : {*nokia.maps.gfx.Graphics*}

This method scales the current projection matrix.

set (key, value) : {*nokia.maps.gfx.Graphics*}

This method changes an IDL state property named by the caller.

setIdl (idl) : {*nokia.maps.gfx.Graphics*}

This method replaces the current IDL with the object supplied by the caller.

stroke () : {*nokia.maps.gfx.Graphics*}

This method strokes the current path.

Methods

strokeText (text, x, y, nx, ny) : {[nokia.maps.gfx.Graphics](#)}

This method creates text path at the given position, oriented by a direction vector provided by the caller, closes it and then strokes it with the stroke color.

translate (dx, dy) : {[nokia.maps.gfx.Graphics](#)}

This method translates the current projection matrix.

Class Description

The class represents a two-dimensional graphic context which produces an IDL (Image Description Language) object that can be rendered into a real image, using an instance of [nokia.maps.gfx.Painter](#). The graphics context itself controls an IDL state and uses it to generate and IDL object.

Note that after creating a graphics context, the method `beginImage()` must be called, except when the graphics context has been initialized with a previously created IDL, for example:

```
// Initialize local variables.
var Color = nokia.maps.gfx.Color,
    parseCss = Color.parseCss,
    body = document.body || document.documentElement,
    graphics,
    painter,
    image,
    imageNode;

// Create a new image with a size of 64x64 pixels.
graphics = new nokia.maps.gfx.Graphics();
graphics.beginImage(64, 64, "This is my image");
graphics.drawRect(16,16, 31,31);
graphics.set("fillColor", parseCss("red",0.5));
graphics.set("strokeColor", parseCss("black"));
graphics.set("lineWidth", 1);
graphics.fill();
graphics.stroke();
image = graphics.getIDL();

// Create a DOM element from the image using the default painter.
painter = new nokia.maps.gfx.Painter.defaultPainter();
imageNode = painter.createElement(image, document);

// Add this image node to the document.
body.appendChild(imageNode);
```

The above example creates a square box with the dimensions 32 x 32 pixels and a 1 pixel wide black border, filled with semi-transparent red color. Each side of the box measures 32 pixels and the inner square (inside the border) is 30 x 30 pixels. The graphics context uses the same coordinate projection as the IDL itself. A path consists of points and each point is infinitely small. The default origin of the coordinate system is the center of the top-left pixel of the destination surface

here

(image). This means that, by default, the coordinate (0,0) is projected to the center of the top-left pixel at the position (0,0) of the image being created. Therefore if `lineWidth` is set to 1 and a `moveTo(1, 1) . lineTo(200, 1)` is executed, all the pixels from the position (1,1) to (200,1) are stroked with a one-pixel-wide brush. Because the line is infinitely small and runs through the center of each pixel, all the pixels under the pen are colored, resulting in a line that is 201 pixels long.

Repeating the same exercise, but with `lineWidth` of two pixels, results in a 203 pixel long and 3 pixel high line, where the pixels from (1,1) to (200,1) are fully filled with black color. In the pixel row from (0,0) to (202,1) and from (0,2) to (202,2) only half of each pixel is filled. This is because we use a two-pixel-wide brush through the center of each pixel from (1,1) to (200,1), which means that we have to stroke one pixel above and one pixel below the center of those pixels. Therefore from the center of the pixel (1,1) we stroke one pixel up, filling the top half of the pixel (1,1) and the bottom half of the pixel (1,0). We then stroke one pixel down, filling the bottom half of the pixel (1,1) and the top half of the pixel (1,2). This results in a completely filled pixel (1,1), while the pixel (1,0) and (1,2) are only half filled. As it is impossible in reality to fill half a pixel, the rendering engine simulates this effect by not making the pixel fully black, but by calculating a color value that takes half the current color and half the new color taking opacity into account. This can lead to unwanted effects, therefore we advice cuation with `lineWidth` values that are multiples of 2.

Hint: For a box with a specific outer or inner size, subtract the line width from or add it to the length of the side of the box. So if you want a box with an outer size of exactly 200 pixels and a border width of 3 pixels, reduce the width and height in the call to `drawRect ( )` by three, which means you need to call the method, specifying 197 as width and height. This results in a rectangle with an outer size of 200 x 200 pixels and an inner size of 194 x 194 pixels, with a border that is three pixels wide.

Constructor Details

`nokia.maps.gfx.Graphics(idl)`

This method creates a new graphics context, either initialized with an IDL or an empty context.

Parameters:

`idl`: `{nokia.maps.gfx.IDL}` [optional]
An instance of IDL with which to initialize the new graphics context

Method Details

`beginImage(width, height, description): {nokia.maps.gfx.Graphics}`

This method resets the state of the graphics context and starts a new 2D image. The method must be called after the graphics context has been created.

here

Parameters:

width:	{Number}	
		The width of the image to be created
height:	{Number}	
		The height of the image
description:	{String} [optional]	
		A description of the image, if omitted "undefined" is used

Returns:

{[nokia.maps.gfx.Graphics](#)}

The reference to the graphics context (this)

beginPath(): {[nokia.maps.gfx.Graphics](#)}

This method clears all sub-paths and starts a new path. The pen position is lost and a 'moveTo' operation has to be executed to ensure that there is a sub-path with cursor position (x,y) to continue drawing. Therefore the user agent must check if the context has any sub-path. In the absence of a sub-path, the user agent must create a one with the coordinates (x,y) as its first (and only) point, as if the 'moveTo' method had been called.

Returns:

{[nokia.maps.gfx.Graphics](#)}

A reference to the given graphics context (this)

bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y): {[nokia.maps.gfx.Graphics](#)}

This method draws a Bezier curve from the current cursor position to a new position.

Note that the method does not stroke or fill the path.

Parameters:

cp1x: {Number}

here

	The x component of the Bezier curve's first control point
cp1y:	{Number}
	The y component of the Bezier curve's first control point
cp2x:	{Number}
	The x component of the Bezier curve's second control point
cp2y:	{Number}
	The y component of the Bezier curve's second control point
x:	{Number}
	The x component of the new pixel position
y:	{Number}
	The y component of the new pixel position

Returns:

{[nokia.maps.gfx.Graphics](#)}

A reference to the given graphics context (this)

closePath(): {[nokia.maps.gfx.Graphics](#)}

This method closes the current sub-path.

Returns:

{[nokia.maps.gfx.Graphics](#)}

A reference to the given graphics context (this)

drawEllipse(x, y, w, h): {[nokia.maps.gfx.Graphics](#)}

This method creates a new ellipse path.

Note that the method does not stroke or fill the path.

Parameters:

here

x:	{Number}	
		The left edge of the ellipse
y:	{Number}	
		The top edge of the ellipse
w:	{Number}	
		The width of the ellipse
h:	{Number}	
		The height of the ellipse

Returns:

`{nokia.maps.gfx.Graphics}`
A reference to the given graphics context (this)

drawRect(x, y, w, h, rx, ry): `{nokia.maps.gfx.Graphics}`

This method creates a new rectangle sub-path and closes it.

Note that the operation does not cause the path to be stroked or filled. If you want to stroke the path, you should be aware that the path itself does not have a width or height, therefore a rectangle with the top-left corner at 0,0 and a size of 100 x 100 units becomes a 101 pixels wide rectangle if stroked with a `lineWidth` of 1. Stroking of the same original rectangle with a `lineWidth` of 3 produces a rectangle whose width is 103 pixels.

Parameters:

x:	{Number}	
		The left edge of the rectangle
y:	{Number}	
		The top edge of the rectangle
w:	{Number}	
		The width of the rectangle

here

h: {Number}

The height of the rectangle

rx: {Number} [optional]

The horizontal radius of rounded corners. If omitted it will indicate no border rounding

ry: {Number} [optional]

The vertical radius of rounded corners. If not specified it will be the same as rx

Returns:

{[nokia.maps.gfx.Graphics](#)}

A reference to the given graphics context (this)

fill(): {[nokia.maps.gfx.Graphics](#)}

This method fills the current path.

Returns:

{[nokia.maps.gfx.Graphics](#)}

A reference to the given graphics context (this)

fillText(text, x, y, nx, ny): {[nokia.maps.gfx.Graphics](#)}

This method creates text path at the given position, oriented by a direction vector provided by the caller, closes it and then fills it with the fill color. Use [nokia.maps.gfx.Graphics#strokeText](#) method to add glow to text.

Parameters:

text: {String}

The text to draw.

x: {Number}

here

	The x component of the text anchor point
y:	{Number}
	The y component of the text anchor point
nx:	{Number} [optional]
	The x component of the absolute direction vector endpoint; if not provided, x + 1 is used
ny:	{Number} [optional]
	The y component of the absolute direction vector endpoint; if not provided, y is used

Returns:

`{nokia.maps.gfx.Graphics}`

A reference to the given graphics context (this)

getDescription(): {String}

This method retrieves the description of the image.

Returns:

{String} The description of the image

getHeight(): {Number}

This method retrieves the height of the currently rendered image.

Returns:

{Number} The height of the currently rendered image

getIDL(): {[nokia.maps.gfx.IDL](#)}

This method retrieves the IDL of the given graphics context.

Returns:

here

`{nokia.maps.gfx.IDL}`

The IDL of the given graphics context

getWidth(): {Number}

This method retrieves the width of the currently rendered image.

Returns:

{Number} The width of the currently rendered image

lineTo(x, y): {*nokia.maps.gfx.Graphics*}

This method draws a line from the current position to the pen position specified by the caller.

Note that the method does not stroke or fill the path.

Parameters:

x: {Number}
 The x-offset for the final pen position

y: {Number}
 The y-offset for the final pen position

Returns:

{*nokia.maps.gfx.Graphics*}
 A reference to the given graphics context (this)

moveTo(x, y): {*nokia.maps.gfx.Graphics*}

This method moves the cursor to the given position and starts a new sub-path.

Parameters:

x: {Number}
 The x-offset for the cursor position

here

y: {Number}

The y-offset for the cursor position

Returns:

{[nokia.maps.gfx.Graphics](#)}

A reference to the given graphics context (this)

polylineTo(coords): {[nokia.maps.gfx.Graphics](#)}

This method draws a polyline from the current position along all the path specified by the caller as an array of locations.

Note that the method does not stroke or fill the path.

Parameters:

coords: {Number []}

An array of alternating x- and y-components of coordinates relative to the current position.

Returns:

{[nokia.maps.gfx.Graphics](#)}

A reference to the given graphics context (this)

quadraticCurveTo(cpx, cpy, x, y): {[nokia.maps.gfx.Graphics](#)}

This method draws a quadratic curve from the current cursor position to a new position.

Note that the method does not stroke or fill the path.

Parameters:

cpx: {Number}

The x component of the quadratic curve's control point

cpy: {Number}

The y component of the quadratic curve's control point

here

x: {Number}
The x component of the new pixel position

y: {Number}
The y component of the new pixel position

Returns:

{[nokia.maps.gfx.Graphics](#)}
A reference to the given graphics context (this)

restore(): {[nokia.maps.gfx.Graphics](#)}

This method restores a previously saved state of the graphic context or, if the state has not been saved, replaces the current state with the default state.

If the internal stack is empty, the method restores the default state.

Returns:

{[nokia.maps.gfx.Graphics](#)}
A reference to the given graphics context (this)

rotate(angle): {[nokia.maps.gfx.Graphics](#)}

This method rotates the current projection matrix (z axis rotation).

Note that this does rotates the matrix of the context, rather than the image itself, which causes any subsequently created paths to be rotated, without affecting existing paths.

Parameters:

angle: {Number}
The rotation angle in degrees

Returns:

{[nokia.maps.gfx.Graphics](#)}
A reference to the given graphics context (this)

here

save(): `{nokia.maps.gfx.Graphics}`

This method saves the current state of the graphics context on the internal stack.

Returns:

`{nokia.maps.gfx.Graphics}`

The reference to the given graphics context (*this*)

scale(sx, sy): `{nokia.maps.gfx.Graphics}`

This method scales the current projection matrix.

Note that the operation scales the matrix of the context, rather than the image itself, which causes any subsequently created paths to be scaled, without affecting existing paths.

Parameters:

sx: `{Number}`

The scale factor along the x axis

sy: `{Number}`

The scale factor along the y axis

Returns:

`{nokia.maps.gfx.Graphics}`

A reference to the given graphics context (*this*)

set(key, value): `{nokia.maps.gfx.Graphics}`

This method changes an IDL state property named by the caller. The properties that can be changed are: width, height, description, strokeColor, fillColor, lineWidth, lineCap, lineJoin, miterLimit, font and textAlign.

Note that changing the width or height properties results in a complete reset of the graphics state and clearing of the image buffer, therefore changing the size clears and resets the graphics context.

Parameters:

here

key: {String}
The name of the property to modify

value: {Object}
The value to which to set the property

Returns:
{[nokia.maps.gfx.Graphics](#)}
A reference to the given graphics context (this)

See: [nokia.maps.gfx.IDL](#)

setIDL(idl): {[nokia.maps.gfx.Graphics](#)}

This method replaces the current IDL with the object supplied by the caller. The change of IDL affects the image size and description, but has no effect on the saved states.

Parameters:

idl: {[nokia.maps.gfx.IDL](#)}
An object representing the IDL to load

Returns:
{[nokia.maps.gfx.Graphics](#)}
A reference to the given graphics context (this)

Throws:

{String} An exception is thrown if the caller provides an illegal argument that does not represent a valid instance of `nokia.maps.gfx.IDL`

stroke(): {[nokia.maps.gfx.Graphics](#)}

This method strokes the current path.

Returns:

here

`{nokia.maps.gfx.Graphics}`

A reference to the given graphics context (this)

strokeText(text, x, y, nx, ny): `{nokia.maps.gfx.Graphics}`

This method creates text path at the given position, oriented by a direction vector provided by the caller, closes it and then strokes it with the stroke color.

Parameters:

text: `{String}`

The text to draw

x: `{Number}`

The x component of the text anchor point

y: `{Number}`

The y component of the text anchor point

nx: `{Number}` [optional]

The x component of the absolute direction vector endpoint; if not provided, x + 1 is used

ny: `{Number}` [optional]

The y component of the absolute direction vector endpoint; if not provided, y is used

Returns:

`{nokia.maps.gfx.Graphics}`

A reference to the given graphics context (this)

translate(dx, dy): `{nokia.maps.gfx.Graphics}`

This method translates the current projection matrix.

here

Note that this does translates the matrix of the context, rather than the image itself, which causes any subsequently created paths to be translated, without affecting existing paths.

Parameters:

dx: {Number}
The translation distance along the x axis

dy: {Number}
The translation distance along the y axis

Returns:

{[nokia.maps.gfx.Graphics](#)}

A reference to the given graphics context (this)

Class: GraphicsImage

This class is a member of [nokia.maps.gfx](#).

Extends: [nokia.maps.gfx.Image](#)

Class Summary

The class GraphicsImage is designed to render images within the client application at run time.

[For full details, see [nokia.maps.gfx.GraphicsImage](#)]

Table 114: Property Summary

Properties
readonly isGraphics : {Boolean}
This property indicates if an image is an instance of GraphicsImage (true) or not (false).

Directly Inherited Properties

Inherited from class [nokia.maps.gfx.Image](#) :

[height](#), [opacity](#), [state](#), [width](#)

here

Table 115: Method Summary

Methods
<i>getIdl</i> () : {<i>nokia.maps.gfx.IDL</i>} This method retrieves the IDL object created by the given instance of graphics context.

Directly Inherited Methods

Inherited from class *nokia.maps.gfx.Image* :

clone, *createElement*, *getDocument*, *prepare*, *setOpacity*

Class Description

This class renders images within the client application at run time. The graphics image is not rendered at construction time for the same reason that a bitmap image is not loaded at construction time: rendering occurs when it is really necessary. Imagine there are 1000 map objects and the image for each object must be rendered depending on the properties of the object, for example, because the object contains traffic information. It would cost large amounts of CPU time to render all the images and consume a lot of memory to keep the image data in memory while perhaps only a few of the objects may be visible in the current view.

The rendering of a graphics image is done by a function that is not called as soon as the image is needed. The same function can be used for multiple graphics images, therefore it is possible to implement a multi-painter, so a function that renders all markers and uses the marker properties as a source of information about how to render the markers.

It is also possible to create an instance of GraphicsImage without a rendering function, but using a pre-created *nokia.maps.gfx.IDL* or *nokia.maps.gfx.Graphics* context.

```
// Create a few shortcuts.
var GraphicsImage = nokia.maps.gfx.GraphicsImage,
    Color = nokia.maps.gfx.Color,
    parseCss = Color.parseCss;

// Create a reference to the document body.
var body = document.body || document.documentElement;

// Create a new graphics image.
var image = new GraphicsImage(function (graphics, graphicsImage) {
    // This will clear the canvas and set it's size to 64 x 64 pixel
    graphics.beginImage(64, 64, "MyImage");

    // Let's draw a circle with a 2 pixel black border and fill it with semi
    // transparent red color
    graphics.set("lineWidth", 2 );
    graphics.set("strokeColor", parseCss("black"));
});
```

here


```
graphics.set("fillColor", parseCss("red", 0.5));

// Create the path using a helper function.
graphics.drawEllipse(16,16, 32,32);

// Stroke and fill the path.
graphics.stroke();
graphics.fill();
});

// Add the graphics image to the body.
body.appendChild(image.createElement());
```

Constructor Details

nokia.maps.gfx.GraphicsImage(render_fn, context, doc, painter, args)

This method creates a new graphics image instance.

Parameters:

render_fn: {Function | [nokia.maps.gfx.IDL](#) | [nokia.maps.gfx.Graphics](#)}

A function to be called as soon as the image is required; the function gets as its first argument the graphics context (an instance of [nokia.maps.gfx.Graphics](#)) that it uses to render the image, and as a second parameter an instance of [nokia.maps.gfx.GraphicsImage](#), which represents the image to render; if the constructor is called with additional arguments, these are forwarded as additional arguments to the rendering function; optionally, an instance of [nokia.maps.gfx.IDL](#) or [nokia.maps.gfx.Graphics](#) context can be supplied to create a new graphics image instantly

context: {Object} [optional]

The context in which the rendering function is to be executed; if not specified the context (this) is bound to the given instance of [nokia.maps.gfx.GraphicsImage](#)

doc: {Document} [optional]

The document to which the given graphics image is bound; if omitted, the image is bound to the current document

painter: {[nokia.maps.gfx.Painter](#)} [optional]

here

The painter to be used to paint; if not supplied, the painter currently assigned to the static property `defaultPainter` of the class [nokia.maps.gfx.Painter](#) is used; if this argument has an invalid value, it is ignored and the `defaultPainter` is used

`args ...:` `{Object}` [optional]

An arbitrary number of additional arguments to be forwarded to the rendering function

Property Details

readonly **isGraphics:** `{Boolean}`

This property indicates if an image is an instance of `GraphicsImage` (`true`) or not (`false`).

Default Value: `true`

Method Details

getIdl(): `{nokia.maps.gfx.IDL}`

This method retrieves the IDL object created by the given instance of graphics context.

Returns:

`{nokia.maps.gfx.IDL}`

The IDL generated by the given instance of graphics context

Class: IDL

This class is a member of [nokia.maps.gfx](#).

Class Summary

This class represents an image described in intermediate language.

[For full details, see [nokia.maps.gfx.IDL](#)]

Table 116: Property Summary

Properties	
static	attributeToIdentifier : <code>{Object}</code>

here

Properties

This hashmap allows a fast translation from an attribute name (key) to the attribute identifier (value).

data: {Object[]}

This property holds an array of objects, each containing an IDL opcodes and associated parameters; the opcodes and their parameters describe the image.

readonly **description:** {String}

This property holds the description text of the image described by the given IDL instance.

readonly **fillColor:** {Number}

This property holds the current fill color as a 32-bit integer, default is 255 (black, transparent).

readonly **font:** {String}

This property holds the current font specified as a CSS font declaration, default is "10px sans-serif".

readonly **height:** {Number}

This property holds the height of the image described by the given IDL instance.

static **identifierToAttribute:** {Object}

This hashmap allows a fast translation from an attribute identifier (key) to the attribute name (value).

readonly **lineCap:** {String}

This property holds the current line cap, can be "butt", "round" or "square", the default is "butt".

readonly **lineJoin:** {String}

This property holds the current line join, can be "round", "bevel" or "miter", the default is "miter".

readonly **lineWidth:** {Number}

This property holds the current line width in pixel, defaults is 1.

readonly **miterLimit:** {Number}

This property holds the current miter limit, the default is 10.

readonly **opacity:** {String}

This property holds the global opacity is applied to all shapes and combined with the stroke- and fill-opacity to generate the final opacity of a shape.

static **opcodes:** {Object}

This property holds a hashtable that contains opcodes, their numeric values, and descriptions.

readonly **strokeColor:** {Number}

This property holds the current stroke color as a 32-bit integer, default is 0 (black, opaque).

readonly **textAlign:** {String}

This property holds the current text alignment, can be "start", "end", "left", "right" or "center", default is "start".

readonly **width:** {Number}

This property holds the width of the image described by the given IDL.

here

Table 117: Method Summary

Methods
append (idl) : {<i>nokia.maps.gfx.IDL</i>} This method appends the IDL supplied by the caller to the current IDL instance, without changing the size or state of the received IDL.
clone () : {<i>nokia.maps.gfx.IDL</i>} This method returns a clone of the given IDL instance.
concat (idl) : {<i>nokia.maps.gfx.IDL</i>} This method combines the given IDL instance with the one supplied by the caller and returns a merged IDL.
push (args) This method pushes the given IDL commands to the data array and it applies the matrix to all coordinates provided as parameters of opcodes.
resetState () This method resets the state of the given IDL instance object, and therefore sets all the properties to their defaults, clears the stack, but does not modify the width, height, description and data properties.
restoreState () : {Object} This method restores all attributes, except for width, height, description and data, from an internal stack.
saveState (additionalData) This method saves all attributes, except for width, height, description and data, to an internal stack.

Class Description

This class represents an image described in terms of a set of IDL (Image Description Language) instructions known as opcodes. The opcodes are interpreted to render the image based on a virtual rendering context that supports, in addition to opcodes, a set of state attributes. The opcodes either modify the state attributes or they cause drawing operations to be performed, using the current rendering context state.

The coordinate system

A point in IDL is infinitely small and a line or a Bezier curve are therefore infinitely thin as well. An IDL point with the coordinates 0,0 is projected onto the center of the screen pixel with the coordinates 0,0. This projection model differs from that used in SVG and the Canvas implementation, but it fits in with the model used by VML. We settled on this model simply because it makes rendering easier to understand and to handle.

The following example creates a 2D image, 64 x 64 pixels, showing a semi-transparent red rectangle, centered within the image and with a 1 pixel black opaque border:

here

```
// Create a new IDL object.
var IDL = nokia.maps.gfx.IDL,
    parseCss = nokia.maps.gfx.Color.parseCss,
    top = IDL.opcodes,
    attr = IDL.attributeToIdentifier,
    idlImage = new IDL(
[
    // Create a new image with a size of 64x64 pixel
    op.BEGIN_2D_IMAGE, 64, 64, "MyImage",

    // Create a new path that follows a rectangular shape with the top-left
    // corner of the bounding box at 16, 16 and a width and height of 32 x 32
    // pixels
    op.BEGIN_PATH,
    op.MOVE_TO, 16, 16,
    op.LINE_TO, 47, 16,
    op.LINE_TO, 47, 47,
    op.LINE_TO, 16, 47,
    op.CLOSE_PATH,

    // Fill and stroke the path
    op.SET, attr.fillColor, parseCss("red", 0.5),
    op.SET, attr.strokeColor, parseCss("black", 1.0),
    op.SET, attr.lineWidth, 1.0,
    op.FILL,
    op.STROKE
]);

// Create an instance of the best painter for the current browser and
// platform and paint the IDL image into the document body
var body = document.body || document.documentElement,
    DefaultPainter = nokia.maps.gfx.Painter.defaultPainter,
    painter = new DefaultPainter();
body.appendChild(painter.createElement(idlImage, document));
```

Note that we do not recommend that you create an IDL object directly; it is better to use the class [nokia.maps.gfx.Graphics](#) for this purpose. The reason is that although "hand optimized" opcode can be small and maybe efficient in certain implementations, for example, a Canvas painter, but it can cause suboptimal performance if used with other painters such as VML, while the graphics context generates code that runs well in all implementations.

Constructor Details

nokia.maps.gfx.IDL(idl)

This method creates a new empty image description language object.

Parameters:

idl: [{nokia.maps.gfx.IDL}](#) [optional]

here

The IDL with which the given instance of the class is to be initialized; if not provided, an empty class instance is created

Property Details

static attributeToIdentifier: {Object}

This hashmap allows a fast translation from an attribute name (key) to the attribute identifier (value).

data: {Object[]}

This property holds an array of objects, each containing an IDL opcodes and associated parameters; the opcodes and their parameters describe the image.

The data always consists of one opcode, followed by fixed number of parameters for that opcode.

readonly description: {String}

This property holds the description text of the image described by the given IDL instance.

readonly fillColor: {Number}

This property holds the current fill color as a 32-bit integer, default is 255 (black, transparent).

See: [nokia.maps.gfx.Color](#)

readonly font: {String}

This property holds the current font specified as a CSS font declaration, default is "10px sans-serif".

readonly height: {Number}

This property holds the height of the image described by the given IDL instance.

static identifierToAttribute: {Object}

This hashmap allows a fast translation from an attribute identifier (key) to the attribute name (value).

here

readonly **lineCap**: {String}

This property holds the current line cap, can be "butt", "round" or "square", the default is "butt".

The lineCap attribute defines the type of endings of lines. The butt value means that the end of each line has a flat edge. The round value means that a semi-circle is added to the end of the line. The square value means that a rectangle shall be added at the end of each line.

readonly **lineJoin**: {String}

This property holds the current line join, can be "round", "bevel" or "miter", the default is "miter".

The lineJoin defines the type of corners that will be places where two lines meet. If the lineJoin is "bevel", a filled triangle connecting the two opposite corners of the straight line, with the third point of the triangle being the join point.

The "round" value means that a filled arc connecting the two aforementioned corners of the join must be rendered at joins.

The "miter" value means that a second filled triangle must be rendered at the join, with one line being the line between the two aforementioned corners, abutting the first triangle, and the other two being continuations of the outside edges of the two joining lines, as long as required to intersect without going over the miter length.

The miter length is the distance from the point where the join occurs to the intersection of the line edges on the outside of the join. The miter limit ratio is the maximum allowed ratio of the miter length to half the line width. If the miter length would cause the miter limit ratio to be exceeded, this second triangle must not be rendered.

readonly **lineWidth**: {Number}

This property holds the current line width in pixel, defaults is 1.

readonly **miterLimit**: {Number}

This property holds the current miter limit, the default is 10.

See: [nokia.maps.gfx.IDL#lineCap](https://nokia.maps.gfx.idl#lineCap)

readonly **opacity**: {String}

here

This property holds the global opacity is applied to all shapes and combined with the stroke- and fill-opacity to generate the final opacity of a shape. The value must be in the range from 0.0 (fully transparent) to 1.0 (opaque).

`static opcodes: {Object}`

This property holds a hashtable that contains opcodes, their numeric values, and descriptions. The contents are represented as list of opcode aliases, where each of the aliases is followed by a further list showing the opcode value, the parameters and a description. The opcode aliases serve as hashtable keys (for example `BEGIN_PATH`), while the values are the numeric values of opcodes. Please use this reference when adding your own opcodes to the `data` property of the IDL.

- `BEGIN_2D_IMAGE`
 - Opcode: 45 (-)
 - Parameters: {Number} **width**, {Number} **height**, {String} **description**
 - Description: This is the first opcode (0x2D) that must appear in any 2D image description. It defines a new image with the given size of the canvas and sets the cursor to the position 0,0.
- `BEGIN_PATH`
 - Opcode: 64 (@)
 - Parameters: -
 - Description: Clear all sub-paths.
- `MOVE_TO`
 - Opcode: 77 (M)
 - Parameters: {Number} **x**, {Number} **y**
 - Description: Start a new subpath and move the cursor to the given pixel position within the canvas.
- `LINE_TO`
 - Opcode: 76 (L)
 - Parameters: {Number} **x**, {Number}
 - Description: Draw a line from the current cursor position to the given canvas position.
- `BEZIER_CURVE_TO`
 - Opcode: 67 (C)
 - Parameters: {Number} **cp1_x**, {Number} **cp1_y**, {Number} **cp2_x**, {Number} **cp2_y**, {Number} **x**, {Number} **y**

here

- Description: Draw a Bezier curve of the 3rd degree from the current cursor position to the given **x** / **y** position using the two given control points.
- CLOSE_PATH
 - Opcode: 120 (x)
 - Parameters: -
 - Description: Close the last sub-path and connect the last point of the path with the first one, so that a fill operation can be performed.
- FILL
 - Opcode: 102 (f)
 - Parameters: -
 - Description: Close the last sub-path and fill all sub-paths with the current fill color.
- STROKE
 - Opcode: 115 (s)
 - Parameters: -
 - Description: Stroke all sub-paths with the current stroke style.
- DRAW_IMAGE
 - Opcode: 73 (I)
 - Parameters: {Image} **image**, {Number} **srcX**, {Number} **srcY**, {Number} **srcWidth**, {Number} **srcHeight**, {Number} **destX**, {Number} **destY**, {Number} **destWidth**, {Number} **destHeight**
 - Description: Slice part of a source image and draws it into the destination.
- DRAW_IDL
 - Opcode: 105 (i)
 - Parameters: {nokia.maps.gfx.IDL} **idl**, {Number} **srcX**, {Number} **srcY**, {Number} **srcWidth**, {Number} **srcHeight**, {Number} **destX**, {Number} **destY**, {Number} **destWidth**, {Number} **destHeight**
 - Description: Slice part of a source IDL and draws it into the destination.
- SAVE
 - Opcode: 62 (>)
 - Parameters: -
 - Description: Save the state of the rendering context at the stack (push).
- RESTORE
 - Opcode: 60 (<)

here

- Parameters: -
- Description: Restore the state of the rendering context from the stack (pop).
- SET
 - Opcode: 35 (#)
 - Parameters: {Number} **attributeIdentifier**, {Object} **value**
 - Description: Set the state attribute with the given identifier to the given value. The name of an attribute can be translated into its identifier using the static `attributeToIdentifier` property and from its identifier to its name using the static `identifierToAttribute` property.

readonly **strokeColor**: {Number}

This property holds the current stroke color as a 32-bit integer, default is 0 (black, opaque).

See: [nokia.maps.gfx.Color](#)

readonly **textAlign**: {String}

This property holds the current text alignment, can be "start", "end", "left", "right" or "center", default is "start".

readonly **width**: {Number}

This property holds the width of the image described by the given IDL.

Method Details

append(idl): {[nokia.maps.gfx.IDL](#)}

This method appends the IDL supplied by the caller to the current IDL instance, without changing the size or state of the received IDL.

Parameters:

idl: [{nokia.maps.gfx.IDL}](#)

The IDL to be appended to the given IDL instance

Returns:

here

`{nokia.maps.gfx.IDL}`

The reference to the given IDL instance

clone(): `{nokia.maps.gfx.IDL}`

This method returns a clone of the given IDL instance.

Note that the only property that is deep-cloned is `data`. As a result, the clone contains a copied instruction set, but all other properties refer to the same values. The internal stack is cloned as well.

Returns:

`{nokia.maps.gfx.IDL}`

A clone of the given IDL instance

concat(idl): `{nokia.maps.gfx.IDL}`

This method combines the given IDL instance with the one supplied by the caller and returns a merged IDL.

Parameters:

`idl`: `{nokia.maps.gfx.IDL}`

The IDL to be appended to the given IDL instance

Returns:

`{nokia.maps.gfx.IDL}`

The new combined IDL, the image size of the current IDL is retained

push(args)

This method pushes the given IDL commands to the data array and it applies the matrix to all coordinates provided as parameters of opcodes.

Example:

```
var IDL = nokia.maps.gfx.IDL,
    Color = nokia.maps.gfx.Color,
    idl = new IDL(),
```

here

```

    opcodes = IDL.opcodes,
    attr = attributeToIdentifier;
idl.push(
  opcodes.BEGIN_2D_IMAGE, 40, 40, "ExampleImage",
  opcodes.BEGIN_PATH,
  opcodes.MOVE_TO, 10, 10,
  opcodes.LINE_TO, 20, 10,
  opcodes.LINE_TO, 20, 20,
  opcodes.LINE_TO, 10, 20,
  opcodes.CLOSE_PATH,
  opcodes.SET, attr.fillColor, Color.parseCss("#ff0000",0.5),
  opcodes.FILL
);

```

Parameters:

args ...: {Object}

An arbitrary number of parameters that contain the IDL code to be pushed

resetState()

This method resets the state of the given IDL instance object, and therefore sets all the properties to their defaults, clears the stack, but does not modify the width, height, description and data properties.

restoreState(): {Object}

This method restores all attributes, except for width, height, description and data, from an internal stack.

If the internal stack is empty, the method restores the default state and therefore it has the same effect as reset().

Returns:

{Object} Additional data that has been pushed to the internal stack, if any

saveState(additionalData)

This method saves all attributes, except for width, height, description and data, to an internal stack.

Parameters:

here

`additionalData:` `{Object}` [optional]

If provided, this object containing additional data is saved together with the rest of the state

Class: Image

This class is a member of [`nokia.maps.gfx`](#).

Class Summary

This class is an base class for all image implementations like graphic or bitmap image.

[For full details, see [`nokia.maps.gfx.Image`](#)]

Table 118: Property Summary

Properties
readonly <code>height</code>: <code>{Number}</code> This property holds the height of the image in pixels.
<code>opacity</code>: <code>{Number}</code> This property holds the default opacity for the image.
readonly <code>state</code>: <code>{Number}</code> This property holds the state of the image.
readonly <code>width</code>: <code>{Number}</code> This property holds the width of the image in pixels.

Table 119: Method Summary

Methods
<code>clone</code> (doc) : <code>{nokia.maps.gfx.Image}</code> This method clones the given image and optionally binds it to a different document.
<code>createElement</code> (opacity) : <code>{Node}</code> This method creates a DOM node containing the given image.
static <code>fromObject</code> (obj, doc) : <code>{nokia.maps.gfx.Image}</code> This method attempts to create an appropriate <code>nokia.maps.gfx.Image</code> object from the object supplied by the caller.
<code>getDocument</code> () : <code>{Document}</code> This method retrieves the document to which the given image is bound.
<code>prepare</code> (callback, context, args) : <code>{nokia.maps.gfx.Image}</code>

here

Methods

This method prepares the given image and calls a callback on completion or failure.

setOpacity (element, opacity) : {*nokia.maps.gfx.Image*}

This method changes the global opacity of an element returned by the `createElement()`.

Class Description

This class is an abstract base class for all kinds of image implementations. An image is immutable: after an image instance has been created, it is prepared for painting, and once it has been prepared, it can no longer be modified. If the `prepare()` method is not called explicitly after an image has been created, the method `createElement()` invokes `prepare()` implicitly. Preparation depends on the image type, for example, the `BitmapImage` class loads the binary data of the image from a Web server, while the class `GraphicsImage` generates an IDL object that makes it possible to render the final image directly within the client, resulting in a VML, SVG or a Canvas element.

Further implementation are possible, each implementation must add a listener to the static array `fromObjectListener` to convert arbitrary objects into instances of the `Image` class automatically.

Note that each image is bound to a document and can only be used with that document. The binding must be set at image construction time and can not be changed later. The reason is that the `prepare()` method call is optional and can be explicit after the construction of the image and before the image is used or it is called implicitly by `createElement()`, which can be invoked on demand. If the document were passed as an argument to `createElement()`, problems might arise for explicitly prepared images as preparation may require knowledge about the document to which the image is to be added.

The following code extract shows how an image can be used in different frames:

```
var someFunction = function (gfxImage) {
  // Ensure that the given image is valid for our
  // current document and if not, import it.
  if (gfxImage.getDocument() !== document)
    gfxImage = gfxImage.clone(document);
}
```

Property Details

readonly **height**: {Number}

This property holds the height of the image in pixels. The value may be zero until the image has been successfully prepared.

Note that the default value is zero, because a calculation with `undefined` would lead to an NaN result

here

opacity: {Number}

This property holds the default opacity for the image. The value is used when `createElement()` is called without a parameter. The value must be between 0 and 1 (default).

readonly **state:** {Number}

This property holds the state of the image.

The list below shows the possible values and explains their meaning:

- `undefined` - the image preparation has not been started yet
- `-2` - the image preparation failed and calling the `prepare` again won't solve the problem
- `-1` - the image preparation failed, but calling the `prepare` again might solve the problem
- `0` - the image preparation is currently ongoing
- `1` - the image preparation is finished successfully

Creating DOM nodes for an unprepared image or an image whose preparation failed results in an empty image with a size of 0 x 0 pixels.

readonly **width:** {Number}

This property holds the width of the image in pixels. The value may be zero until the image has been successfully prepared.

Note that the default value is zero, because a calculation with `undefined` would lead to a NaN result

Method Details

clone(doc): {*[nokia.maps.gfx.Image](#)*}

This method clones the given image and optionally binds it to a different document.

Parameters:

doc: {Document} [optional]

A reference to the document to which to bind the image clone; if provided and different from the document to which the image is currently bound, the clone is bound to the document to which this argument refers

here

Returns:

`{nokia.maps.gfx.Image}`

A reference to the image clone

createElement(opacity): {Node}

This method creates a DOM node containing the given image.

This method is synchronous and returns a rendered DOM node that represents the image. If the image is not yet successfully prepared, the preparation has failed or is ongoing, the returned image may be a blank image, with a size of 0 x 0 pixels. If the image is not yet prepared and no preparation is currently ongoing, this method calls the `prepare()` method.

Note that if an image is being prepared and the returned DOM node has the size of 0 x 0 pixels, the size may be updated automatically as soon as the preparation has finished successfully. Therefore, it is fail-safe to create, for example, a `BitmapImage` and to call the method `createElement()` directly, because if the image has not yet been loaded, it will then be loaded asynchronously and the returned DOM node will be extended as soon as the image is available.

Parameters:

opacity: {Number} [optional]

If provided, the value of this argument represents the opacity of the image being created; valid values lie between 0 (transparent) and 1 (opaque); if the argument is not provided, the value of the image property `opacity` is used

Returns:

{Node} A reference to the DOM node that can be added to the document containing the given image

Example:

```
// Create a bitmap image.
var myImage = new nokia.maps.gfx.BitmapImage(
    "http://www.w3.org/Icons/www/w3c_home_nb.png");

// Add the image to the document, prepare() is called
// by the method createElement().
var body = document.body || document.documentElement;
body.appendChild(myImage.createElement());
```

here

static fromObject(obj, doc): {*nokia.maps.gfx.Image*}

This method attempts to create an appropriate *nokia.maps.gfx.Image* object from the object supplied by the caller.

Parameters:

obj:	{Object}
	An object to be converted into an Image instance
doc:	{Document} [optional]
	The document to which the new Image instance is to be bound; if not provided, the current document is used

Returns:

{ <i>nokia.maps.gfx.Image</i> }
An instance of the Image class created from the received object or null if no Image object can be created

Example:

```
// If GraphicsImage is included in the gfx
// package, it returns a graphics image on
// execution of the following code:
var myImage = nokia.maps.gfx.Image.fromObject(
    '<svg xmlns="http://www.w3.org/2000/svg" width="1000" height="600"
    viewBox="0 0 5 3">'+
    '<rect id="black_stripe" width="5" height="3" y="0" x="0" fill="#000"/>'+
    '<rect id="red_stripe" width="5" height="2" y="1" x="0" fill="#D00"/>'+
    '<rect id="gold_stripe" width="5" height="1" y="2" x="0"
    fill="#FFCE00"/>'+
    '</svg>'
);
```

getDocument(): {Document}

This method retrieves the document to which the given image is bound.

Returns:

here

`{Document}` The document to which the given image is bound (set by the constructor and cannot be modified)

prepare(callback, context, args): `{nokia.maps.gfx.Image}`

This method prepares the given image and calls a callback on completion or failure. The callback and the context in which the callback is called are both provided by the caller.

Calling this method on an image that is already prepared leads to the callback being invoked immediately. If the method is called multiple times, all the supplied callback handlers are called in the order in which the method has been called.

Parameters:

callback: `{Function}`

A function to call as soon as the preparation is finished, whether successfully or not. The function must be able to receive as its first argument a reference to the GFX image, followed by all the additional parameters passed to this method

context: `{Object}` [optional]

An object representing the context in which to call the callback; if not supplied, the context (this) is bound to the given image object.

args ...: `{Object}` [optional]

An arbitrary number of additional arguments to be passed to the callback

Returns:

`{nokia.maps.gfx.Image}`

An instance of `gfx . Image` representing the given image (this)

Example:

```
// Create a new bitmap image.
var myImage = new nokia.maps.gfx.BitmapImage("http://www.w3.org/Icons/WWW/w3c_home_nb.png");

// Create a callback handler that shall be called as
// soon as the image is loaded.
var myCallback = function (image, msg) {
    // If preparing the image failed, show an error
```

here

```
// message and return.
if (image.state<1) return alert ("Failed to load image!");

// Show the message supplied to the prepare method,
// if any.
if (msg) alert (msg);

// Paint the image into the document.
var body = image.doc.body || image.doc.documentElement;
body.appendChild(image.createElement());
};

// Start the image preparation which (in this case)
// loads the bitmap from the Web server and then calls
// the callback. If the image is within the browser's
// cache, it may happen that the callback method is
// called immediately.
myImage.prepare(myCallback, null, "Hello World!");
```

setOpacity(element, opacity): {[nokia.maps.gfx.Image](#)}

This method changes the global opacity of an element returned by the `createElement()`.

Parameters:

element:	{Node}
	The DOM node that was returned by the <code>createElement()</code>
opacity:	{Number}
	The global opacity for the node as a number between 0 and 1

Returns:

{[nokia.maps.gfx.Image](#)}

The reference to the given image object (this)

Class: Painter

This class is a member of [nokia.maps.gfx](#).

Class Summary

This class is an abstract base class for all painter implementations.

[For full details, see [nokia.maps.gfx.Painter](#)]

here

Table 120: Property Summary

Properties
static defaultPainter : {Function}
This property holds a reference to the constructor of the default painter that can be used with the current browser/platform.

Table 121: Method Summary

Methods
createElement (idl, doc, opacity) : {Node}
This method renders the supplied IDL object and returns a DOM node that can be added to the supplied document.
setOpacity (element, opacity)
This method changes the global opacity of an element returned by the method createElement().
setScale (element, scale)
This method changes the scaling of an element returned by the method createElement().
updateElement (idl, node, opacity)
This method updates a DOM node that was previously returned by the method createElement() to the caller-supplied IDL object.

Class Description

This class is an abstract base class for all painter implementations. A painter renders a *nokia.maps.gfx.IDL* object using a specific method, for example via a CANVAS tag (*nokia.maps.gfx.CanvasPainter*), an SVG DOM node (*nokia.maps.gfx.SvgPainter*) or a VML DOM node (*nokia.maps.gfx.VmlPainter*).

Property Details

static **defaultPainter**: {Function}

This property holds a reference to the constructor of the default painter that can be used with the current browser/platform.

Note that the default painter need not to be the best painter, but it must be a painter that is fully supported by the current browser/platform. It may happen that there is no default painter, so this property may be set to null.

Method Details

createElement(idl, doc, opacity): {Node}

here

This method renders the supplied IDL object and returns a DOM node that can be added to the supplied document. The node must contain "something" (VML, SVG, Canvas tag or whatever is suitable) that can hold the image described by the IDL.

Parameters:

idl:	<code>{<i>nokia.maps.gfx.IDL</i>}</code>
	An IDL object describing the image to be rendered
doc:	<code>{Document}</code> [optional, default: document]
	A reference to the document for which the painter is to create an element; if omitted the current document is used
opacity:	<code>{Number}</code> [optional, default: 1]
	A value representing the opacity with which the element is to be created; if omitted or invalid, 1.0 (opaque) is used; must be a number between 0 (transparent) and 1 (opaque)

Returns:

<code>{Node}</code>	A reference to the DOM node that can be added into the given DOM node and represents the image described by the supplied IDL

setOpacity(element, opacity)

This method changes the global opacity of an element returned by the method `createElement()`.

Parameters:

element:	<code>{Node}</code>
	A reference to the DOM node that was returned by <code>createElement()</code>
opacity:	<code>{Number}</code>
	A value representing global opacity for the node as a number between 0 and 1

setScale(element, scale)

here

This method changes the scaling of an element returned by the method `createElement()`.

Parameters:

element:	<code>{Node}</code>
	A reference to the DOM node that was returned by <code>createElement()</code>
scale:	<code>{<i>nokia.maps.util.Point</i>}</code>
	A value representing scale factor for x and y axes

updateElement(idl, node, opacity)

This method updates a DOM node that was previously returned by the method `createElement()` to the caller-supplied IDL object. This can result in a complete re-rendering of the node's content or in a partial update, depending on the painter implementation and the number of changes made to the IDL.

Parameters:

idl:	<code>{<i>nokia.maps.gfx.IDL</i>}</code>
	An IDL object describing the image to be rendered
node:	<code>{Node}</code>
	A reference to the DOM node that was previously returned by the method <code>createElement()</code> and which is to be updated
opacity:	<code>{Number}</code> [optional, default: 1]
	A numeric value representing the opacity with which the element is to be updated; if omitted, or an invalid value, 1.0 (opaque) is used; the value must be a number between 0 (transparent) and 1 (opaque)

Class: SvgPainter

This class is a member of `nokia.maps.gfx`.

Extends: `nokia.maps.gfx.Painter`

here

Class Summary

This class renders IDL into SVG tags.

[For full details, see [nokia.maps.gfx.SvgPainter](#)]

Table 122: Method Summary

Methods
createMarkup (idl) : {String} This method creates an SVG mark-up string from the received instance of nokia.maps.gfx.IDL .

Directly Inherited Methods

Inherited from class [nokia.maps.gfx.Painter](#) :

[createElement](#), [setOpacity](#), [setScale](#), [updateElement](#)

Class Description

The class [SvgPainter](#) renders a [nokia.maps.gfx.IDL](#) into a DOM node with an SVG tag. This painter works with all browsers that support SVG. To instantiate this class, use the default constructor without arguments:

```
var mySvgPainter= new nokia.maps.gfx.SvgPainter();
```

Constructor Details

[nokia.maps.gfx.SvgPainter\(\)](#)

This method creates a new [SvgPainter](#) instance.

Method Details

[createMarkup](#)(idl): {String}

This method creates an SVG mark-up string from the received instance of [nokia.maps.gfx.IDL](#).

Parameters:

idl: {[nokia.maps.gfx.IDL](#)}

An image description language object that is to be converted into a mark-up string.

here

Returns:

{String} The SVG mark-up string

Class: SvgParser

This class is a member of [nokia.maps.gfx](#).

Class Summary

This class converts a SVG string into IDL language.

[For full details, see [nokia.maps.gfx.SvgParser](#)]

Table 123: Method Summary

Methods
parseSvg (svgMarkup) : {nokia.maps.gfx.IDL} This method parses the supplied SVG string into an IDL object.
parseSvgInfo (svgMarkup) : {nokia.maps.gfx.SvgParser.SvgMarkupInfo} This method returns information about the SVG mark-up.
parseSvgToGraphics (svgMarkup, graphics) : {nokia.maps.gfx.Graphics} This method parses an SVG string and renders it using a graphics context.

Class Description

This class provides a facility to convert an SVG string into an instance of [nokia.maps.gfx.IDL](#).

Example:

```
// Parse an SVG mark-up string.
var svgParser = new nokia.maps.gfx.SvgParser(),
    svgText = '<svg xmlns="http://www.w3.org/2000/svg" version="1.1"
width="28" height="36" viewBox="0 0 28 36">' +
    '<path stroke="none" fill-opacity="0.40000003799999995" fill="#000000"
d="M 18.9 33.4 C 18.9 34.8 16.7 36 13.899999999999999 36 C
11.099999999999998 36 8.899999999999999 34.8 8.899999999999999
33.4 C 8.899999999999999 32 11.099999999999998 30.799999999999997
13.899999999999999 30.799999999999997 C 16.7 30.799999999999997 18.9 31.9
18.9 33.4 Z">' +
    '</path><path stroke="none" fill-opacity="1" fill="#666666" d="M 14 0 C
6.3 0 0 6.3 0 14.1 C 0 16.2 0.4 18.2 1.3 20.1 L 1.8 21.1 C 3 23.1 12.3 32.2
13.3 33.2 L 14 33.9 L 14.7 33.2 C 15.7 32.2 25 23.1 26.1 21.2 L 26.7 20.2 C
27.5 18.2 28 16.2 28 14.1 C 28 6.3 21.7 0 14 0 Z">' +
    '</path><path stroke="none" fill-opacity="1" fill="#666666" d="M 14 0.9 C
21.2 0.9 27 6.8 27 14 C 27 16 26.6 17.9 25.8 19.6 L 25.3 20.5 C 24.2 22.4
```

here


```

14 32.4 14 32.4 C 14 32.4 3.8 22.4 2.7 20.5 L 2.2 19.6 C 1.4 17.9 1 16 1 14
C 1 6.8 6.8 0.9 14 0.9 Z">' +
'</path><path stroke="none" fill-opacity="1" fill="#ffffff" d="M 14 0.9 C
6.8 0.9 1 6.8 1 14 C 1 16 1.4 17.9 2.2 19.6 L 2.7 20.5 C 3.8 22.4 14 32.4
14 32.4 C 14 32.4 24.2 22.4 25.3 20.5 L 25.8 19.6 C 26.6 17.9 27 16 27 14 C
27 6.8 21.2 0.9 14 0.9 Z">' +
'</path><path stroke="none" fill-opacity="1" fill="#ffffff" d="M 14 1.9 C
20.6 1.9 26 7.3 26 14 C 25.9 16.1 25.5 18.2 24.4 20 C 23.8 21.2 17.9 27.1
14 30.9 C 10.1 27.1 4.2 21.2 3.6 20.1 C 2.6 18.2 1.9 16.2 2 14 C 2 7.3 7.4
1.9 14 1.9 Z">' +
'</path><path stroke="none" fill-opacity="1" fill="#43a51b" d="M 14 1.9
C 7.4 1.9 2 7.3000000000000001 2 14 C 1.9 16.2 2.6 18.2 3.6 20.1 C 4.2
21.3 10.1 27.1 14 31 C 17.9 27.1 23.8 21.3 24.4 20.1 C 25.5 18.3 25.9
16.2000000000000003 26 14.1000000000000001 C 26 7.4000000000000001 20.6
2.00000000000000018 14 2.00000000000000018 L 14 2.00000000000000018 Z">' +
'</path><path stroke="none" fill-opacity="0.7019608509999999"
fill="#ffffff" d="M 21 6.1 C 17.8 3.5 13.7 2.5 11.9 3.9 C 11.4 4.3 11.2 4.8
11.1 5.4 C 15.2 6 18.2 8.1000000000000001 19.6 9.9 C 22 13 21.6 13.9 21.6
13.9 C 22.4000000000000002 13.9 23 13.7000000000000001 23.5 13.3 C 25.3 11.9
24.2 8.8 21 6.2000000000000001 L 21 6.2000000000000001 Z">' +
'</path><text font-family="arial" font-size="13px" font-weight="bold"
alignment-baseline="middle" text-anchor="middle" fill="#ffffff"
opacity="0.866666749" y="0" x="0" transform="matrix(1, 0, 0, 1, 14,
15)">X</text></svg>',
    idlImage = svgParser.parseSvg(svgText);

// And show it within the page.
var image = new nokia.maps.gfx.GraphicsImage(idlImage),
    body = document.body || document.documentElement;
body.appendChild(image.createElement());

```

Note that this parser does not support the full SVG standard but [SVG Tiny 1.1 standard](#).

Constructor Details

nokia.maps.gfx.SvgParser()

This method creates a new instance of `SvgParser`

Method Details

parseSvg(svgMarkup): {*nokia.maps.gfx.IDL*}

This method parses the supplied SVG string into an IDL object.

Parameters:

svgMarkup: {String | Document}
An SVG mark-up string or an XML document

here

Returns:

`{nokia.maps.gfx.IDL}`

An IDL generated from the SVG mark-up

parseSvgInfo(svgMarkup): `{nokia.maps.gfx.SvgParser.SvgMarkupInfo}`

This method returns information about the SVG mark-up.

Parameters:

svgMarkup: `{String | Document}`

An SVG mark-up string or an XML document

Returns:

`{nokia.maps.gfx.SvgParser.SvgMarkupInfo}`

Information about the SVG mark-up or null if the SVG parsing failed

parseSvgToGraphics(svgMarkup, graphics): `{nokia.maps.gfx.Graphics}`

This method parses an SVG string and renders it using a graphics context. The method can be used to scale SVGs, as shown in the example below:

```
// Create the SVG parser, a graphics context and the SVG string
// that defines a 100 x 100 pixels SVG, with a circle in the
// middle that has a radius of 40 pixels.
var gfx = nokia.maps.gfx,
    svgParser = new gfx.SvgParser(),
    svgString = '<svg width="100" height="100" xmlns:xlink="http://
www.w3.org/1999/xlink" xmlns="http://www.w3.org/2000/svg"><g><title>Layer
1</title><circle stroke-width="5" fill="#FF0000" id="svg_1" r="40" cy="50"
cx="50"/></g></svg>',
    originalImage,
    scaledImage,
    body = document.getElementsByTagName("BODY")[0];

// First let's create a graphics image in original size.
originalImage = new gfx.GraphicsImage(
    // This function is called when the image is to be rendered.
    function (graphics) {
        // Parse the SVG string into an IDL.
        var idl = svgParser.parseSvg(svgString);

        // Create an image from the IDL.
```

here

```

        graphics.setIDL(idl);
    }
};

// And now one with the size doubled.
scaledImage = new gfx.GraphicsImage(
    // This method is called when the image is to be rendered.
    function (graphics) {
        // Parse the SVG and query information about it.
        var svgInfo = svgParser.parseSvgInfo(svgString);

        // Create a new image of the double size.
        graphics.beginImage(svgInfo.width*2, svgInfo.height*2, "scaled
image");

        // Scale x- and y-axis by 2.
        graphics.scale(2,2);

        // Render the SVG string into the existing graphics context using
        // the already parsed SVG document (this is faster than
        // parsing the XML twice).
        svgParser.parseSvgToGraphics(svgInfo.document, graphics);
    }
);

// Render the images into the document.
body.appendChild(originalImage.createElement());
body.appendChild(scaledImage.createElement());

```

Parameters:

svgMarkup: {String | Document}

An SVG mark-up string or an XML document.

graphics: {[nokia.maps.gfx.Graphics](#)} [optional]

The graphics context that is to be used for rendering; if omitted, a new context is created and then returned

Returns:

{[nokia.maps.gfx.Graphics](#)}

The graphics context that was used to render this SVG

Interface: [SvgMarkupInfo](#)

This interface is a member of [nokia.maps.gfx.SvgParser](#).

here

Interface Summary

This is an abstract interface for documentation purpose only.

[For full details, see [nokia.maps.gfx.SvgParser.SvgMarkupInfo](#)]

Table 124: Property Summary

Properties
<i>document</i>: {Document} This property holds the XML document containing the definition of the SVG image.
<i>height</i>: {Number} This property holds the height of the SVG image in pixels.
<i>width</i>: {Number} This property holds the width of the SVG image in pixels.

Interface Description

This is an abstract interface for documentation purpose only. The class provides the information returned by the method [nokia.maps.gfx.SvgParser#parseSvgInfo\(\)](#).

Property Details

document: {Document}

This property holds the XML document containing the definition of the SVG image.

height: {Number}

This property holds the height of the SVG image in pixels.

width: {Number}

This property holds the width of the SVG image in pixels.

Class: VmlPainter

This class is a member of [nokia.maps.gfx](#).

Extends: [nokia.maps.gfx.Painter](#)

here

Class Summary

This class renders a IDL into VML tags.

[For full details, see [nokia.maps.gfx.VmlPainter](#)]

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.gfx.Painter](#) :

[createElement](#), [setOpacity](#), [setScale](#), [updateElement](#)

Class Description

The class VmlPainter renders an instance of [nokia.maps.gfx.IDL](#) into a VML tag to be used with Internet Explorer. To instantiate this class, use the default constructor without arguments:

```
var myVmlPainter= new nokia.maps.gfx.VmlPainter();
```

Constructor Details

nokia.maps.gfx.VmlPainter()

This method creates a new CanvasPainter instance

Namespace: heatmap

This namespace is a member of [nokia.maps](#).

Namespace Summary

This namespace contains resources that allow the API user to create and manage heat maps.

Namespace Description

This namespace contains resources that allow the API user to create and manage heat maps.

Class: Overlay

This class is a member of [nokia.maps.heatmap](#).

Extends: [nokia.maps.map.provider.Provider](#)

here

Class Summary

This class creates heat map overlay attachable to *nokia.maps.map.Display*.

[For full details, see *nokia.maps.heatmap.Overlay*]

Table 125: Property Summary

Properties
readonly <i>assumeValues</i> : {Boolean} This property holds a boolean value defining whether to paint assumed values in regions where no data is available.
readonly <i>coarseness</i> : {Number} This property holds a numeric value defining the resolution reduction when producing tiles.
readonly <i>colors</i> : { <i>nokia.maps.heatmap.Overlay.Colors</i> } This property holds a <i>nokia.maps.heatmap.Overlay.Colors</i> object that defines the colors of the heat map.
readonly <i>opacity</i> : {Number} This property holds the opacity of the heat map overlay.
readonly <i>sampleDepth</i> : {Number} This property holds a numeric value defining the number of sampling iterations the heat map renderer will perform on the data set.
readonly <i>type</i> : {String} This property holds the type of the heat map to be generated.

Directly Inherited Properties

Inherited from class *nokia.maps.map.provider.Provider* :

description, *getInvalidationMark*, *id*, *label*, *max*, *min*

Table 126: Method Summary

Methods
<i>addData</i> (data) This method adds an array of <i>nokia.maps.heatmap.Overlay.DataPoints</i> to the given heat map.
<i>clear</i> () This method removes all data from the given heat map provider.

Directly Inherited Methods

Inherited from class *nokia.maps.map.provider.Provider* :

getCopyrights, *providesLevel*, *shutdown*, *update*

here

Inherited from class [nokia.maps.util.EventTarget](#) :

[addListener](#), [dispatch](#), [removeListener](#)

Event Summary

Directly Inherited Events

Inherited from class [nokia.maps.map.provider.Provider](#) :

[response](#), [update](#)

Class Description

This class creates tiles to visualize value-based or density-based heat maps. User can choose between density and value based heat map.

Constructor Details

nokia.maps.heatmap.Overlay(options)

This method creates an empty heat map overlay.

Parameters:

options: {[nokia.maps.heatmap.Overlay.Options](#)} [optional]

An object specifying settings for the new heat map overlay instance

Property Details

readonly **assumeValues**: {Boolean}

This property holds a boolean value defining whether to paint assumed values in regions where no data is available. This is especially useful for value maps which are generated from a small data sets as tiles with no available data will be filled with the next available average value in the data set.

The property is read only and can be set at construction time as part of the [nokia.maps.heatmap.Overlay.Options](#) initialization properties.

Default Value: false

readonly **coarseness**: {Number}

here

This property holds a numeric value defining the resolution reduction when producing tiles. If the coarseness is set to 0 tiles will be produced in the original resolution. A coarseness of 1 allows the renderer to render tiles at half the size and then scale the output, a coarseness of 2 allows the renderer to create tiles at a size of a quarter of the original tile size. Increasing the number dramatically increases performance but also reduces visual quality, especially when using posterization (non-interpolated colors). Values may range between 0 and 3.

The property is read only and can be set at construction time as part of the [nokia.maps.heatmap.Overlay.Options](#) initialization properties.

Default Value: 1

readonly **colors**: {[nokia.maps.heatmap.Overlay.Colors](#)}

This property holds a [nokia.maps.heatmap.Overlay.Colors](#) object that defines the colors of the heat map.

The property is read only and can be set at construction time as part of the [nokia.maps.heatmap.Overlay.Options](#) initialization properties.

readonly **opacity**: {Number}

This property holds the opacity of the heat map overlay. Values may range from 0+ to 1, inclusive.

The property is read only and can be set at construction time as part of the [nokia.maps.heatmap.Overlay.Options](#) initialization properties.

Default Value: 0.8

readonly **sampleDepth**: {Number}

This property holds a numeric value defining the number of sampling iterations the heat map renderer will perform on the data set. Each iteration will sample the data more finely. Higher values will create more detailed maps but also cost performance. Values may range between 1 and 8.

The property is read only and can be set at construction time as part of the [nokia.maps.heatmap.Overlay.Options](#) initialization properties.

Default Value: 4

readonly **type**: {String}

here

This property holds the type of the heat map to be generated.

The property is read only and can be set at construction time as part of the [nokia.maps.heatmap.Overlay.Options](#) initialization properties.

The possible values are:

- "density"
- "value"

Default Value: "density"

Method Details

addData(data)

This method adds an array of [nokia.maps.heatmap.Overlay.DataPoints](#) to the given heat map. The heat map provides a visual representation of these data.

Note that there is a significant performance cost to adding and removing data, once an overlay has been created. This is related to the rendering engine and tile caching. If you wish to add data after tiles have been created, the cache must be cleared and all tiles need to be recreated. For this reason, heat maps should not be used to create fast animations.

Parameters:

data: {[nokia.maps.heatmap.Overlay.DataPoint](#)[]}

The array of data points to be added to the overlay.

Example:

```
// Create a density heat map overlay:
var myHeatMapOverlay = new nokia.maps.heatmap.Overlay();

// Add DataPoints:
myHeatMapOverlay.addData([
    {latitude: 50, longitude: 13},
    {latitude: 50, longitude: 13.1},
    {latitude: 51, longitude: 13.2},
    {latitude: 49.9, longitude: 13.15},
    ...
    {latitude: 50, longitude: 13.4}
]);

// Add the overlay to the map display to make
// the heat map visible:
myDisplay.overlays.add(myHeatMapOverlay);
```

here

clear()

This method removes all data from the given heat map provider. New data can be provided using the [nokia.maps.heatmap.Overlay#addData](#) method.

Interface: Colors

This interface is a member of [nokia.maps.heatmap.Overlay](#).

Interface Summary

This interface defines a standard way to customize the colors displayed in a heat map by associating custom colors with normalized "heat" values.

[For full details, see [nokia.maps.heatmap.Overlay.Colors](#)]

Table 127: Property Summary

Properties
interpolate: {Boolean} This property is a flag indicating whether interpolation is to be used to display smooth color transitions in the heat map (true) or whether the heat map is to be posterized (false). stops: {Object} This property holds a map, where the keys are numbers ranging from 0 to 1, inclusive, and the corresponding values are the colors defined in the hexadecimal Web color notation, including opacity.

Interface Description

This interface defines a standard way to customize the colors displayed in a heat map by associating custom colors with normalized "heat" values.

The interface associates colors with "heat" values via the "stops" property. This property is a map, where the keys are numbers ranging from 0 to 1, inclusive, and the values are colors associated with these numeric values. The colors are represented using CSS colors, i.e. hexadecimal notation ("RGB" or "RRGGBB") or rgb(a) notation ("rgba(R, G, B, A)"), etc. The range (0 .. 1) stands for the normalized "heat" value. Points on the heat map are colorized using the color-stops information from the Colors object.

Property Details

interpolate: {Boolean}

here

This property is a flag indicating whether interpolation is to be used to display smooth color transitions in the heat map (`true`) or whether the heat map is to be posterized (`false`).

stops: {Object}

This property holds a map, where the keys are numbers ranging from 0 to 1, inclusive, and the corresponding values are the colors defined in the hexadecimal Web color notation, including opacity.

Interface: *DataPoint*

This interface is a member of [*nokia.maps.heatmap.Overlay*](#).

Interface Summary

This interface represents a single data point that can be visualized in a heat map.

[For full details, see [*nokia.maps.heatmap.Overlay.DataPoint*](#)]

Table 128: Property Summary

Properties
latitude: {Number} This property defines the latitude of this data point.
longitude: {Number} This property defines the longitude of this data point.
value: {Number} This property defines a numeric "heat" value associated with the given data point.

Interface Description

This interface represents a single data point that can be visualized in a heat map. It consists of the geographic coordinates defining its position and an optional value to be associated with this position.

Property Details

latitude: {Number}

This property defines the latitude of this data point.

longitude: {Number}

here

This property defines the longitude of this data point.

value: {Number}

This property defines a numeric "heat" value associated with the given data point. If no value is defined, the default is 1.

Default Value: 1

Interface: Options

This interface is a member of [nokia.maps.heatmap.Overlay](#).

Interface Summary

This interface defines options that can be passed to the [nokia.maps.heatmap.Overlay](#) constructor.

[For full details, see [nokia.maps.heatmap.Overlay.Options](#)]

Table 129: Property Summary

Properties
assumeValues: {Boolean} This property holds a boolean value defining whether to paint assumed values in regions where no data is available.
coarseness: {Number} This property holds a numeric value defining the resolution reduction when producing tiles.
colors: {nokia.maps.heatmap.Overlay.Colors} This property holds an object that conforms to the interface nokia.maps.heatmap.Overlay.Colors and is used to define the colors of the heat map.
opacity: {Number} This property holds the opacity of the heat map overlay.
sampleDepth: {Number} This property holds a numeric value defining the number of sampling iterations the heat map renderer will perform on the data set.
type: {String} This property specifies the type of the heat map to be generated.

Directly Inherited Properties

Inherited from class [nokia.maps.map.provider.Provider.Options](#) :

here

[description](#), [getCopyrights](#), [label](#), [max](#), [min](#), [updateCycle](#)

Interface Description

This interface defines options that can be passed to the [nokia.maps.heatmap.Overlay](#) constructor.

Property Details

assumeValues: {Boolean}

This property holds a boolean value defining whether to paint assumed values in regions where no data is available. This is especially useful for value maps which are generated from a small data sets as tiles with no available data will be filled with the next available average value in the data set.

Default Value: false

coarseness: {Number}

This property holds a numeric value defining the resolution reduction when producing tiles. If the coarseness is set to 0 tiles will be produced in the original resolution. A coarseness of 1 allows the renderer to render tiles at half the size and then scale the output, a coarseness of 2 allows the renderer to create tiles at a size of a quarter of the original tile size. Increasing the number dramatically increases performance but also reduces visual quality, especially when using posterization (non-interpolated colors). Values may range between 0 and 3.

Default Value: 1

colors: {[nokia.maps.heatmap.Overlay.Colors](#)}

This property holds an object that conforms to the interface [nokia.maps.heatmap.Overlay.Colors](#) and is used to define the colors of the heat map. The property is optional. If it is not provided, the following default color definition is used:

```
{
  stops: {
    "0": "#008", // dark blue
    "0.2": "#0b0", // medium green
    "0.5": "#ff0", // yellow
    "0.7": "#f00" // red
  },
  interpolate: true
}
```

here

This means that the highest "heat" value is displayed in opaque red, the lowest as opaque blue, while values in between are shown in opaque green and yellow and interpolation is used to determine them.

opacity: {Number}

This property holds the opacity of the heat map overlay. Values may range from 0+ to 1, inclusive. The property is optional.

Default Value: 0.8

sampleDepth: {Number}

This property holds a numeric value defining the number of sampling iterations the heat map renderer will perform on the data set. Each iteration will sample the data more finely. Higher values will create more detailed maps but also cost performance. Values may range between 1 and 8.

Default Value: 4

type: {String}

This property specifies the type of the heat map to be generated. The property is optional.

The possible values are:

- "density"
- "value"

Default Value: "density"

Namespace: kml

This namespace is a member of *nokia.maps*.

Namespace Summary

This namespace defines classes and methods that implement support for KML, including import of KML files.

here

Namespace Description

This namespace defines classes and methods that implement support for KML, including import of KML files.

Class: BalloonStyle

This class is a member of [nokia.maps.kml](#).

Class Summary

This class represents the tag <BalloonStyle> defined in the KML specification.

[For full details, see [nokia.maps.kml.BalloonStyle](#)]

Table 130: Property Summary

Properties
readonly displayMode : {String} This property holds the default value for the balloon display mode.
readonly text : {String} This property holds the default value for text displayed in the balloon.

Directly Inherited Properties

Inherited from class [nokia.maps.kml.ColorStyle](#) :

[color](#), [colorMode](#)

Class Description

This class represents the tag <BalloonStyle> defined in the KML specification. Class specifies how the balloon containing text, etc. is drawn.

Constructor Details

nokia.maps.kml.BalloonStyle(node)

The class constructor initializes a new instance of the class, using the arguments supplied by the caller.

Parameters:

node: {Node}

here

An XML node from the KML document that is being parsed

Property Details

readonly **displayMode**: {String}

This property holds the default value for the balloon display mode. If the property is set to "default", the balloon displays text. If `displayMode` is "hide", the balloon remains hidden.

readonly **text**: {String}

This property holds the default value for text displayed in the balloon. You can add entities to the `text` tag using the following format to refer to a child element of `Feature`: `$(name)`, `$(description)`.

For example, in the following KML excerpt, the fields `$(name)` and `$(description)` are replaced by the fields `name` and `description` found in the `Feature` elements that use the given `BalloonStyle`:

```
<text>This is $(name), whose description is:<br />
$(description)</text>
```

Class: ColorStyle

This class is a member of [nokia.maps.kml](#).

Class Summary

This class represents the abstract `<ColorStyle>` tag from the KML specification.

[For full details, see [nokia.maps.kml.ColorStyle](#)]

Table 131: Property Summary

Properties
readonly color : {String} This property holds the color and opacity (alpha) values expressed in hexadecimal notation.
readonly colorMode : {String} This property holds the value of color mode, which can be "normal" (no effect) or "random".

Class Description

This class represents the abstract `<ColorStyle>` tag from the KML specification.

here

Constructor Details

nokia.maps.kml.ColorStyle(node)

The constructor for this class initializes a new instance of the class, using the arguments supplied by the caller.

Parameters:

node:	{Node}
	XML A node from the KML document that is being parsed

Property Details

readonly **color**: {String}

This property holds the color and opacity (alpha) values expressed in hexadecimal notation. The range of values for any one color is 0 to 255 (00 to ff). The order of expression is #rrggbbaa, where aa=alpha (00 to ff); bb=blue (00 to ff); gg=green (00 to ff); rr=red (00 to ff). For alpha, 00 is fully transparent and ff is fully opaque.

readonly **colorMode**: {String}

This property holds the value of color mode, which can be "normal" (no effect) or "random". A value of "random" applies a random linear scale to the base color as follows:

- To achieve a truly random selection of colors, specify white as the base color, #ffffff.
- If you specify a single color component (for example, a value of #ff0000ff for red), random color values for that one component (red) are subsequently selected, ranging from 00 (black) to ff (full red).
- If you specify values for two or for all three color components, a random linear scale is applied to each color component, with results ranging from black to the maximum values specified for each component.
- The opacity of a color comes from the alpha component and is never randomized.

Class: Container

This class is a member of [nokia.maps.kml](#).

Class Summary

This class represents the abstract <Container> tag from the KML specification.

here

[For full details, see [nokia.maps.kml.Container](#)]

Table 132: Property Summary

Properties
readonly features : {Array}
This property holds an array of nokia.maps.kml.Feature elements

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.kml.ObservableNode](#) :

[load](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This class represents the abstract <Container> tag from the KML specification. It is designed to hold other objects and allows for creation of nested hierarchies. Properties are filled asynchronously after call from ObservableNode class during XML tree parse.

Constructor Details

nokia.maps.kml.Container(node, styleContainer, featureStyle)

The constructor initializes a new instance of the class with the arguments supplied by the caller.

Parameters:

node:	{Node}
	An XML node from the KML document which is being parsed
styleContainer:	{ nokia.maps.kml.StyleContainer }
	A container holding a list of <code>StyleSelector</code> objects, which is a list of all global styles that can be referenced by id
featureStyle:	{ nokia.maps.kml.FeatureStyle }
	A container holding locally defined styles that can be applied to the given object

here

Property Details

readonly **features**: {Array}

This property holds an array of *nokia.maps.kml.Feature* elements

Namespace: DOM

This namespace is a member of *nokia.maps.kml*.

Namespace Summary

This is a static class that provides helper methods for the KML-specific classes.

Table 133: Method Summary

Methods
static <i>getAltitudeMode</i> (node) : {String} This method parses the value of an XML DOM node into a string.
static <i>getBoolean</i> (node) : {Boolean} This method parses the value of an XML DOM node to a Boolean value.
static <i>getColor</i> (node) : {String} This method parses the value of an XML DOM node to a string that represents a color in RGBA format, reversing the order, because the KML specification uses the ABGR format.
static <i>getFloat</i> (node) : {Float} This method parses the value of an XML DOM node to a Floating point number.
static <i>getInteger</i> (node) : {Integer} This method parses the value of an XML DOM node to integer.
static <i>getNodeAttribute</i> (node, attributeName) : {String} This method parses the attribute attributeName from an XML DOM node and returns it as a string.
static <i>getNodeId</i> (node) : {String} This method parses the attribute id from an XML DOM node and returns it as a string.
static <i>getString</i> (node) : {String} This method parses the value of a DOM node into string.
static <i>parseCoord</i> (coordStr) : { <i>nokia.maps.geo.Coordinate</i> } This method parses a object containing geographical coordinates from a string.
static <i>parseCoords</i> (coordsString) : { <i>nokia.maps.geo.Coordinate</i> []} This method parses a string containing coordinates to an array of objects, each containing the properties "lat", "lng" and "alt".

here

Methods

```
static parseFromString (node) : {Document}
```

This method parses an XML string into browser-independent XML Document object.

Namespace Description

This is a static class that provides helper methods for the KML-specific classes.

Method Details

```
static getAltitudeMode(node): {String}
```

This method parses the value of an XML DOM node into a string. The resulting string contains only predefined values as specified in KML documentation

Parameters:

node: {Node}
Any XML DOM node to parse

Returns:

{String} One of the predefined string values

```
static getBoolean(node): {Boolean}
```

This method parses the value of an XML DOM node to a Boolean value.

Parameters:

node: {Node}
An XML DOM node to parse

Returns:

{Boolean} A Boolean representation of the node value

```
static getColor(node): {String}
```

here

This method parses the value of an XML DOM node to a string that represents a color in RGBA format, reversing the order, because the KML specification uses the ABGR format. If parsing fails, the method returns the default hard-coded color.

Parameters:

node: {Node}
An XML DOM node to parse

Returns:

{String} A string representing a color in #RGBA format

static **getFloat**(node): {Float}

This method parses the value of an XML DOM node to a Floating point number.

Parameters:

node: {Node}
An XML DOM node to parse

Returns:

{Float} The value of the node as a float

static **getInteger**(node): {Integer}

This method parses the value of an XML DOM node to integer.

Parameters:

node: {Node}
An XML DOM node to parse

Returns:

{Integer} The node value as an integer

static **getNodeAttribute**(node, attributeName): {String}

here

This method parses the attribute `attributeName` from an XML DOM node and returns it as a string.

Parameters:

<code>node:</code>	<code>{Node}</code>
	An XML DOM node
<code>attributeName:</code>	<code>{String}</code>
	Name of the attribute

Returns:

<code>{String}</code>	A string containing the nod id

static **getNodeId**(node): `{String}`

This method parses the attribute `id` from an XML DOM node and returns it as a string.

Parameters:

<code>node:</code>	<code>{Node}</code>
	An XML DOM node

Returns:

<code>{String}</code>	A string containing the node id

static **getString**(node): `{String}`

This method parses the value of a DOM node into string.

Parameters:

<code>node:</code>	<code>{Node}</code>
	A DOM node to parse

Returns:

<code>{String}</code>	The value of the DOM node as a stripped string value

```
static parseCoord(coordStr): {nokia.maps.geo.Coordinate}
```

This method parses a object containing geographical coordinates from a string. If the input does not contain valid coordinates, the return value is undefined.

Parameters:

coordStr: {String | String[]}

A string containing the values of geographical coordinate as specified in KML specification, for example: "13,53.5,0" or "13,53.5" Or an Array of strings like ["13", "53.5", "0"] or ["13", "53.5"]

Returns:

{*nokia.maps.geo.Coordinate*}

An object containing an object containing geographical coordinates (the properties are "lat", "lng", and "alt")

```
static parseCoords(coordsString): {nokia.maps.geo.Coordinate[]}
```

This method parses a string containing coordinates to an array of objects, each containing the properties "lat", "lng" and "alt". The return value contains only valid coordinates.

Parameters:

coordsString: {String}

A string containing a list of coordinates as specified in KML specification, for example: "13,53.5,0 14,53,0 13,52.5,0 14,52.5,0" or non standard comma separated list of coordinates like "13,53.5,0,14,53,0,13,52.5,0,14,52.5,0". In this case specifying altitude(3rd component in each tuple) is required. Please do not use spaces in case if you provide coordinates in 2nd format.

Returns:

{*nokia.maps.geo.Coordinate*[]}

A list of objects containing parsed geographical coordinates

```
static parseFromString(node): {Document}
```

here

This method parses an XML string into browser-independent XML Document object.

Parameters:

node: {String}
 An XML string to parse

Returns:

{Document} An XML DOM document

Class: Document

This class is a member of [nokia.maps.kml](#).

Class Summary

This class represents the <Document> tag from the KML specification.

[For full details, see [nokia.maps.kml.Document](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.kml.Container](#) :

[features](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.kml.ObservableNode](#) :

[load](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This class represents the <Document> tag from the KML specification.

here

Constructor Details

nokia.maps.kml.Document(node, styleContainer, featureStyle)

The constructor initializes a new instance of the class with the arguments supplied by the caller.

Parameters:

node:	{Node}	An XML node from the KML document which is currently being parsed
styleContainer:	{ nokia.maps.kml.StyleContainer }	A list of StyleSelector objects containing all global styles which can be referenced by id
featureStyle:	{ nokia.maps.kml.FeatureStyle }	A list of locally defined styles which can be applied to the given object

Class: Feature

This class is a member of [nokia.maps.kml](#).

Extends: [nokia.maps.kml.ObservableNode](#)

Class Summary

This class represents the abstract <Feature> tag from KML specification.

[For full details, see [nokia.maps.kml.Feature](#)]

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.kml.ObservableNode](#) :

[load](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

here

Class Description

This class represents the abstract <Feature> tag from KML specification. Properties are filled asynchronously after call from ObservableNode class during XML tree parse.

Constructor Details

nokia.maps.kml.Feature(node, styleContainer, featureStyle)

The constructor initializes a new instance of the class with the arguments supplied by the caller.

Parameters:

node: {Node}

An XML node from a KML document that is being parsed.

styleContainer: {[nokia.maps.kml.StyleContainer](#)}

A list of StyleSelector objects, contains all global styles which can be referenced by id

featureStyle: {[nokia.maps.kml.FeatureStyle](#)}

An object containing locally defined styles that can be applied to the given object

Class: FeatureStyle

This class is a member of [nokia.maps.kml](#).

Class Summary

This class represents a style applicable to the class Feature.

[For full details, see [nokia.maps.kml.FeatureStyle](#)]

Table 134: Method Summary

Methods
mergeStyle (style, highlighted, reverse) This method merges the styles from the received arguments to the current instance.

Class Description

This class represents a style applicable to the class Feature.

here

Constructor Details

nokia.maps.kml.FeatureStyle(style, highlighted)

Initialize a new instance of the class with the following parameters.

Parameters:

style:	{ nokia.maps.kml.Style nokia.maps.kml.FeatureStyle }
	This is the set of styles with to initialize the given instance of the class
highlighted:	{Boolean} [optional]
	A value indicating if the style is for highlighted shapes (true) or not (false); if the value is true, the argument style should be an instance of nokia.maps.kml.Style

Method Details

mergeStyle(style, highlighted, reverse)

This method merges the styles from the received arguments to the current instance.

Parameters:

style:	{ nokia.maps.kml.Style nokia.maps.kml.FeatureStyle }
	An object containing the style to be merged with the given instance of this class.
highlighted:	{Boolean} [optional]
	A value indicating if the style is for highlighted shapes (true) or not (false); if the value is true, the argument style should be an instance of nokia.maps.kml.Style
reverse:	{Boolean}
	Specifies priority of the merging operation.

Class: Folder

This class is a member of [nokia.maps.kml](#).

here

Class Summary

This class represents the <Folder> tag from the KML specification.

[For full details, see [nokia.maps.kml.Folder](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.kml.Container](#) :

[features](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.kml.ObservableNode](#) :

[load](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This class represents the <Folder> tag from the KML specification.

Constructor Details

nokia.maps.kml.Folder(node, styleContainer, featureStyle)

The constructor initializes a new instance of the class, using the arguments provided by the caller.

Parameters:

node: {Node}

An XML node from KML document, which is currently parsed.

styleContainer: {[nokia.maps.kml.StyleContainer](#)}

A list of StyleSelector objects, contains all global styles that can be referenced by id

featureStyle: {[nokia.maps.kml.FeatureStyle](#)}

here

Contains locally defined styles which can be applied to the given object

Class: Geometry

This class is a member of [nokia.maps.kml](#).

Extends: [nokia.maps.kml.ObservableNode](#)

Class Summary

This class represents the abstract <Geometry> tag from the KML specification.

[For full details, see [nokia.maps.kml.Geometry](#)]

Table 135: Property Summary

Properties
readonly weight : {Number}
"Weight" of the geometry.

Table 136: Method Summary

Methods
getCenterCoordinate () : { nokia.maps.geo.Coordinate }
This method calculates the arithmetic average of all coordinates.
getStyle () : { nokia.maps.kml.Style }
Returns the style of the Geometry as a set of properties which can be used while creating equivalent map object

Directly Inherited Methods

Inherited from class [nokia.maps.kml.ObservableNode](#) :

[load](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This class represents the abstract <Geometry> tag from the KML specification.

here

Constructor Details

nokia.maps.kml.Geometry(node)

The constructor initializes a new instance of the class, using the arguments provided by the caller.

Parameters:

node: {Node}

An XML node from KML document, which is currently parsed.

Property Details

readonly **weight**: {Number}

"Weight" of the geometry. This attribute is used to measure how heavy geometries are. For example if it is a polyline with 100 connection points, its weight is 100, and if it is a simple point object, than it has a weight of 1.

Default Value: 0

Method Details

getCenterCoordinate(): {*nokia.maps.geo.Coordinate*}

This method calculates the arithmetic average of all coordinates.

Returns:

{*nokia.maps.geo.Coordinate*}

An instance of *nokia.maps.geo.Coordinate* representing the calculated average

Throws:

{*NotImplementedException*} When trying to create object of this class and call this method.

getStyle(): {*nokia.maps.kml.Style*}

Returns the style of the Geometry as a set of properties which can be used while creating equivalent map object

here

Returns:

`{nokia.maps.kml.Style}`

An object containing style properties to draw the given object on map

Throws:

`{NotImplementedException}` When trying to create object of this class and call this method.

Class: Icon

This class is a member of [nokia.maps.kml](#).

Class Summary

This class represents the <Icon> tag from the KML specification.

[For full details, see [nokia.maps.kml.Icon](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.kml.Link](#) :

[href](#)

Class Description

This class represents the <Icon> tag from the KML specification.

Constructor Details

`nokia.maps.kml.Icon`(node)

The constructor initializes a new instance of the class, using the arguments provided by the caller.

Parameters:

node: `{Node}`

An XML node from the KML document that is being parsed

here

Class: IconStyle

This class is a member of [nokia.maps.kml](#).

Class Summary

This class represents the <IconStyle> tag from the KML specification.

[For full details, see [nokia.maps.kml.IconStyle](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.kml.ColorStyle](#) :

[color](#), [colorMode](#)

Class Description

This class represents the <IconStyle> tag from the KML specification.

Constructor Details

nokia.maps.kml.IconStyle(node)

The constructor initializes a new instance of the class, using the arguments provided by the caller.

Parameters:

node: {Node}

An XML node from the KML document which is currently being parsed.

Class: LabelStyle

This class is a member of [nokia.maps.kml](#).

Class Summary

This class represents the <LabelStyle> tag from the KML specification.

[For full details, see [nokia.maps.kml.LabelStyle](#)]

here

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.kml.ColorStyle](#) :

[color](#), [colorMode](#)

Class Description

This class represents the <LabelStyle> tag from the KML specification.

Constructor Details

nokia.maps.kml.LabelStyle(node)

The constructor initializes a new instance of the class, using the arguments provided by the caller.

Parameters:

node: {Node}

An XML node from the KML document which is currently being parsed

Class: LineString

This class is a member of [nokia.maps.kml](#).

Extends: [nokia.maps.kml.Geometry](#)

Class Summary

This class represents the <LineString> tag from the KML specification.

[For full details, see [nokia.maps.kml.LineString](#)]

Table 137: Property Summary

Properties	
readonly	coordinates : { nokia.maps.geo.Coordinate []}
An array containing nokia.maps.geo.Coordinate objects.	

Directly Inherited Properties

Inherited from class [nokia.maps.kml.Geometry](#) :

[weight](#)

here

Table 138: Method Summary

Methods
<code>getCenterCoordinate</code> () : { <i>nokia.maps.geo.Coordinate</i> null}
This method calculates the arithmetic average of all coordinates.
<code>getStyle</code> () : {Object}
This method gets the styles for the given object.

Directly Inherited Methods

Inherited from class *nokia.maps.kml.Geometry* :

getCenterCoordinate, *getStyle*

Inherited from class *nokia.maps.kml.ObservableNode* :

load

Inherited from class *nokia.maps.util.OObject* :

addObserver, *get*, *remove*, *removeObserver*, *set*

Class Description

This class represents the <LineString> tag from the KML specification.

Constructor Details

`nokia.maps.kml.LineString`(node, lineStyle, balloonStyle)

The constructor initializes a new instance of the class, using the arguments provided by the caller.

Parameters:

node:	{Node}
	An XML node from the KML document, which is currently being parsed
lineStyle:	{ <i>nokia.maps.kml.LineStyle</i> }
	An object representing the line style to use during conversion to a map object
balloonStyle:	{ <i>nokia.maps.kml.BalloonStyle</i> }
	An object representing the balloon style to use during conversion to a map object

here

Property Details

readonly **coordinates**: {[nokia.maps.geo.Coordinate](#)[]}

An array containing [nokia.maps.geo.Coordinate](#) objects.

A KML file must include four or more tuples, each consisting of floating point values for longitude, latitude, and altitude. The altitude component is optional. Do not include spaces within a tuple. The last coordinate must be the same as the first coordinate. Coordinates are expressed in decimal degrees only.

It is important to bear in mind that when parsing of coordinates from a KML file has failed, or fewer than 4 points were specified, this array remains empty.

Method Details

getCenterCoordinate(): {[nokia.maps.geo.Coordinate](#) | null}

This method calculates the arithmetic average of all coordinates. This method contains an implementation of [nokia.maps.kml.Geometry#getCenterCoordinate](#).

Returns:

{[nokia.maps.geo.Coordinate](#)
| null}

An instance of `nokia.maps.geo.Coordinate` representing the calculated average, or null if [nokia.maps.kml.LineString#coordinates](#) was empty

getStyle(): {Object}

This method gets the styles for the given object. This method contains an implementation of [nokia.maps.kml.Geometry#getStyle](#).

Returns:

{Object} An object with properties "color" and "width".

Class: LineStyle

This class is a member of [nokia.maps.kml](#).

Class Summary

This class represents the <LineStyle> tag from the KML specification.

here

[For full details, see [nokia.maps.kml.LineStyle](#)]

Table 139: Property Summary

Properties
static DEFAULT_STROKE_COLOR : {String} Default stroke color for KML line object
static DEFAULT_STROKE_WIDTH : {Number} Default stroke width for KML line object

Directly Inherited Properties

Inherited from class [nokia.maps.kml.ColorStyle](#) :

[color](#), [colorMode](#)

Class Description

This class represents the <LineStyle> tag from the KML specification.

Constructor Details

nokia.maps.kml.LineStyle(node)

The constructor initializes a new instance of the class, using the arguments provided by the caller.

Parameters:

node: {Node}

An XML node from KML document which is currently being parsed

Property Details

static **DEFAULT_STROKE_COLOR**: {String}

Default stroke color for KML line object

Default Value: #000000ff

static **DEFAULT_STROKE_WIDTH**: {Number}

Default stroke width for KML line object

here

Default Value: 1

Class: LinearRing

This class is a member of [nokia.maps.kml](#).

Extends: [nokia.maps.kml.Geometry](#)

Class Summary

This class represents the <LinearRing> tag from the KML specification.

[For full details, see [nokia.maps.kml.LinearRing](#)]

Table 140: Property Summary

Properties
readonly coordinates : { nokia.maps.geo.Coordinate []}
An array containing nokia.maps.geo.Coordinates .

Directly Inherited Properties

Inherited from class [nokia.maps.kml.Geometry](#) :

[weight](#)

Table 141: Method Summary

Methods
getCenterCoordinate () : { nokia.maps.geo.Coordinate null}
This method calculates the arithmetic average of all coordinates.
getStyle () : {Object}
This method gets the styles for the given object.

Directly Inherited Methods

Inherited from class [nokia.maps.kml.Geometry](#) :

[getCenterCoordinate](#), [getStyle](#)

Inherited from class [nokia.maps.kml.ObservableNode](#) :

[load](#)

Inherited from class [nokia.maps.util.OObject](#) :

here

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This class represents the <LinearRing> tag from the KML specification.

Constructor Details

nokia.maps.kml.LinearRing(node, lineStyle, balloonStyle)

The constructor initializes a new instance of the class, using the arguments provided by the caller.

Parameters:

node:	{Node}	An XML node from the KML document which is currently being parsed
lineStyle:	{ nokia.maps.kml.LineStyle }	An object representing the line style to use during conversion to a map object
balloonStyle:	{ nokia.maps.kml.BalloonStyle }	An object representing the balloon style to use during conversion to a map object

Property Details

readonly **coordinates**: {[nokia.maps.geo.Coordinate](#)[]}

An array containing [nokia.maps.geo.Coordinates](#).

A KML file is required to have four or more tuples, each consisting of floating point values for longitude, latitude, and altitude. The altitude component is optional. Do not include spaces within a tuple. The last coordinate must be the same as the first coordinate. Coordinates are expressed in decimal degrees only.

It is important to bear in mind that when parsing of coordinates from a KML file has failed, or fewer than 4 points were specified, this array remains empty.

Method Details

getCenterCoordinate(): {[nokia.maps.geo.Coordinate](#) | null}

here

This method calculates the arithmetic average of all coordinates. This method contains an implementation of [nokia.maps.kml.Geometry#getCenterCoordinate](#).

Returns:

```
{nokia.maps.geo.Coordinate
| null}
```

An instance of `nokia.maps.geo.Coordinate` representing the calculated average, or `null` if [nokia.maps.kml.LinearRing#coordinates](#) was empty

getStyle(): {Object}

This method gets the styles for the given object. This method contains an implementation of [nokia.maps.kml.Geometry#getStyle](#).

Returns:

```
{Object}           An object with properties "color" and "width".
```

Class: Link

This class is a member of [nokia.maps.kml](#).

Class Summary

This class represents the <Link> tag from the KML specification.

[For full details, see [nokia.maps.kml.Link](#)]

Table 142: Property Summary

Properties
readonly href : {String}
A URL (either an HTTP address or a local file specification).

Class Description

This class represents the <Link> tag from the KML specification.

Constructor Details

nokia.maps.kml.Link(node)

here

The constructor initializes a new instance of the class, using the arguments provided by the caller.

Parameters:

node: {Node}
An XML node from the KML document that is currently being parsed

Property Details

readonly href: {String}

A URL (either an HTTP address or a local file specification). Relative URLs can be used in this tag and are evaluated relative to the enclosing KML file.

Class: Manager

This class is a member of *nokia.maps.kml*.

Extends: *nokia.maps.util.OObject*

Class Summary

This class submits XHR requests for KML files and parses a KML document.

[For full details, see *nokia.maps.kml.Manager*]

Table 143: Property Summary

Properties
<i>kmlDocument</i> : { <i>nokia.maps.kml.Document</i> } This property holds the parsed KML document.
<i>path</i> : {String} This property holds the path from which kml documented have been fetched
<i>state</i> : {String} This property holds the state of the request sent by the given manager instance.

Table 144: Method Summary

Methods
<i>parse</i> (doc) This method parses a KML document, using the styles, style maps and placemarks defined in the document to create geometries.

here

Methods

`parseKML` (String)

This method submits an XMLHttpRequest and parses the KML document received in response.

Directly Inherited Methods

Inherited from class [`nokia.maps.util.OObject`](#) :

[`addObserver`](#), [`get`](#), [`remove`](#), [`removeObserver`](#), [`set`](#)

Class Description

This class submits XHR requests for KML files and parses a KML document.

Constructor Details

`nokia.maps.kml.Manager` ()

This method initializes a new instance of the class, setting its state property to "initial" and resetting the KML document property to null.

Property Details

`kmlDocument` : {[`nokia.maps.kml.Document`](#)}

This property holds the parsed KML document. You can observe the property `nokia.maps.kml.Manager#state`: if its value is "finished", then `kmlDocument` can be accessed, otherwise `kmlDocument` is undefined.

`path` : {String}

This property holds the path from which kml documented have been fetched

`state` : {String}

This property holds the state of the request sent by the given manager instance. The possible values are: "initial", "started", "loaded" "finished", "failed".

Method Details

`parse`(doc)

here

This method parses a KML document, using the styles, style maps and placemarks defined in the document to create geometries.

Parameters:

doc: {Document}
A KML document that to be parsed

parseKML(String)

This method submits an XMLHttpRequest and parses the KML document received in response.

Parameters:

String:
path A URL link to the KML file

Class: MultiGeometry

This class is a member of [nokia.maps.kml](#).

Extends: [nokia.maps.kml.Geometry](#)

Class Summary

This class represents the <MultiGeometry> tag from KML specification.

[For full details, see [nokia.maps.kml.MultiGeometry](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.kml.Geometry](#) :

[weight](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.kml.Geometry](#) :

[getCenterCoordinate](#), [getStyle](#)

here

Inherited from class [nokia.maps.kml.ObservableNode](#) :

[load](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

The class is a container for zero or more geometry primitives associated with the same feature.

Constructor Details

nokia.maps.kml.MultiGeometry(node, featureStyle, balloonStyle, placemark)

Initialize a new instance of the class with the following parameters.

Parameters:

node:	{Node}
	An XML node from KML document, which is currently parsed
featureStyle:	{ nokia.maps.kml.FeatureStyle }
	A container with locally defined styles which can be applied to the given object
balloonStyle:	{ nokia.maps.kml.BalloonStyle }
	Specifies how the description balloon for placemark is drawn.
placemark:	{ nokia.maps.kml.Placemark }
	Placemark object containing this multigeometry

Class: Object

This class is a member of [nokia.maps.kml](#).

Class Summary

This class represents the abstract <Object> tag from KML specification.

[For full details, see [nokia.maps.kml.Object](#)]

here

Class Description

This class represents the abstract <Object> tag from KML specification.

Constructor Details

nokia.maps.kml.Object(node)

Initialize a new instance of the class with the following parameters.

Parameters:

node: {Node}

An XML node from KML document, which is currently parsed.

Class: ObservableNode

This class is a member of [nokia.maps.kml](#).

Extends: [nokia.maps.util.OObject](#)

Class Summary

This mixin is used for asynchronous parsing of XML elements in [nokia.maps.kml.Feature](#) and [nokia.maps.kml.Geometry](#).

[For full details, see [nokia.maps.kml.ObservableNode](#)]

Table 145: Method Summary

Methods
load (onReadyCallback, parentNode) This method recursively parses a node and all its child nodes.

Directly Inherited Methods

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This class is used as a mixin in [nokia.maps.kml.Feature](#) and [nokia.maps.kml.Geometry](#), and for asynchronous parsing of the XML document containing KML data. Sometimes KML nodes can be very large (for example, a folder node may contain 2000 placemarks). Synchronous parsing of such nodes

here

can cause the "Script running slow..." alert to appear. Thus, nodes that by their nature and according to the [KML OGC specification](#) cannot contain too many child nodes are parsed synchronously (those nodes are called "leaf" nodes, for example <Point> in KML). All other intermediate nodes starting from the <kml> tag in an XML document are parsed asynchronously.

During parsing, for each KML node supported by the Maps API, its equivalent is created. For example, each `Folder` tag is found in a document, results in an instance of the [nokia.maps.kml.Folder](#) class, which is derived from this class, thus it has the ability to parse and create its child tags asynchronously. The parsing procedure starts by calling the `load()` method. Each instance of this class has state, whose value at instantiation is set to `initial`. When `load()` is called for a "leaf" node, its state is set to `ready`. The state of the parent node is set to `ready` when all its child nodes are ready. Thus, when the [nokia.maps.kml.Document](#) class instance representing the <kml> tag from XML has the state "ready", the entire XML document parsing has been completed.

Constructor Details

nokia.maps.kml.ObservableNode(node)

The constructor initializes a new instance of the class using the arguments provided by the caller.

Parameters:

node: {Node}
An XML node from the KML document to parse

Method Details

load(onReadyCallback, parentNode)

This method recursively parses a node and all its child nodes. Parsing uses [nokia.maps.util.Coroutine](#).

Parameters:

onReadyCallback: {Function}
A callback function which is called as soon as the node state is "ready"

parentNode: {Node}
Reference to the parent node

Class: Pair

This class is a member of [nokia.maps.kml](#).

here

Class Summary

This class represents the <Pair> tag from the KML specification.

[For full details, see [nokia.maps.kml.Pair](#)]

Table 146: Method Summary

Methods
<code>getStyle()</code> : { nokia.maps.kml.Style } This method retrieves the styles for the given object.

Class Description

This class represents the <Pair> tag from the KML specification.

Constructor Details

`nokia.maps.kml.Pair`(node, styleContainer)

The constructor initializes a new instance of the class, using the arguments provided by the caller.

Parameters:

node:	{Node}
	An XML node from the KML document that is currently being parsed
styleContainer:	{ nokia.maps.kml.StyleContainer }
	A list of StyleSelector objects, this is a list of all global styles which can be referenced by id

Method Details

`getStyle()`: {[nokia.maps.kml.Style](#)}

This method retrieves the styles for the given object.

Returns:

{ nokia.maps.kml.Style }
A style object, or null if no style is defined

here

Class: Placemark

This class is a member of [nokia.maps.kml](#).

Class Summary

This class represents the <Placemark> tag from the KML specification.

[For full details, see [nokia.maps.kml.Placemark](#)]

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.kml.ObservableNode](#) :

[load](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This class represents the <Placemark> tag from the KML specification.

Constructor Details

nokia.maps.kml.Placemark(node, styleContainer, featureStyle)

The constructor initializes a new instance of the class, using the arguments provided by the caller.

Parameters:

node: {Node}

An XML node from the KML document that is currently being parsed

styleContainer: {[nokia.maps.kml.StyleContainer](#)}

A list of StyleSelector objects, which is a list of all global styles which can be referenced by id

featureStyle: {[nokia.maps.kml.FeatureStyle](#)}

A container with locally defined styles which can be applied to the given object

here

Class: Point

This class is a member of [nokia.maps.kml](#).

Extends: [nokia.maps.kml.Geometry](#)

Class Summary

This class represents the <Point> tag from the KML specification.

[For full details, see [nokia.maps.kml.Point](#)]

Table 147: Property Summary

Properties
readonly coordinate : { nokia.maps.geo.Coordinate } This property holds position of the point.
readonly weight : {Number} Weight of the point geometry is 1, because it represents 1 single coordinate on the map

Directly Inherited Properties

Inherited from class [nokia.maps.kml.Geometry](#) :

[weight](#)

Table 148: Method Summary

Methods
getCenterCoordinate () : { nokia.maps.geo.Coordinate } This method returns an object containing the geographical coordinates of a point.
getStyle () : {Object} This method retrieves the styles for the given object.

Directly Inherited Methods

Inherited from class [nokia.maps.kml.Geometry](#) :

[getCenterCoordinate](#), [getStyle](#)

Inherited from class [nokia.maps.kml.ObservableNode](#) :

[load](#)

Inherited from class [nokia.maps.util.OObject](#) :

here

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This class represents the `<Point>` tag from the KML specification.

Constructor Details

`nokia.maps.kml.Point`(node, iconStyle, iconStyle, balloonStyle)

The constructor initializes a new instance of the class, using the arguments provided by the caller.

Parameters:

node:	<code>{Node}</code>	An XML node from the KML document that is currently being parsed
iconStyle:	<code>{nokia.maps.kml.IconStyle}</code>	An object representing the icon style to be used during conversion to a map object
iconStyle:	<code>{nokia.maps.kml.IconStyle}</code>	An object representing the style for a highlighted icon; to be used to highlight the object when we mouse over them
balloonStyle:	<code>{nokia.maps.kml.BalloonStyle}</code>	An object representing the balloon style to use during conversion to a map object

Property Details

readonly **`coordinate`**: `{nokia.maps.geo.Coordinate}`

This property holds position of the point.

In a KML file, please specify only a single tuple consisting of floating point values for longitude, latitude, and altitude (in that order). Longitude and latitude values are in degrees, where:

- longitude ≥ -180 and ≤ 180
- latitude ≥ -90 and ≤ 90
- altitude values (optional) are in meters above sea level

here

Note that you should not include spaces between the property name-value pairs when defining an instance of `nokia.maps.geo.Coordinate`.

readonly **weight**: {Number}

Weight of the point geometry is 1, because it represents 1 single coordinate on the map

Method Details

getCenterCoordinate(): {[nokia.maps.geo.Coordinate](#)}

This method returns an object containing the geographical coordinates of a point. If the KML file contains more than one coordinate tuple, then first one is returned. This method implements [nokia.maps.kml.Geometry#getCenterCoordinate](#).

Returns:

{[nokia.maps.geo.Coordinate](#)}

returns An object containing the geographical coordinates of a point converted from the received string or undefined if the conversion finishes unsuccessfully

getStyle(): {Object}

This method retrieves the styles for the given object. This method implements [nokia.maps.kml.Geometry#getStyle](#).

Returns:

{Object}

An object whose properties reflect the style attributes set on the given object; the properties may include "scale", "offsetX", "offsetY", "offsetXUnits", "offsetYUnits", "icon", "offsetXHighlighted", "offsetYHighlighted", "offsetXHighlightedUnits", "offsetYHighlightedUnits", "iconHighlighted"

Class: PolyStyle

This class is a member of [nokia.maps.kml](#).

Class Summary

This class represents the <PolyStyle> tag from KML specification.

here

[For full details, see [nokia.maps.kml.PolyStyle](#)]

Table 149: Property Summary

Properties
readonly fill : {Boolean} This property holds a Boolean value which specifies whether to fill the polygon.
readonly outline : {Boolean} This property holds a Boolean value which specifies whether to draw the outline of the polygon.

Directly Inherited Properties

Inherited from class [nokia.maps.kml.ColorStyle](#) :

[color](#), [colorMode](#)

Class Description

This class represents the <PolyStyle> tag from KML specification.

Constructor Details

nokia.maps.kml.PolyStyle(node)

The constructor initializes a new instance of the class, using the arguments provided by the caller.

Parameters:

node: {Node}

An XML node from the KML document that is currently being parsed.

Property Details

readonly **fill**: {Boolean}

This property holds a Boolean value which specifies whether to fill the polygon. By default it is `true`.

readonly **outline**: {Boolean}

This property holds a Boolean value which specifies whether to draw the outline of the polygon. By default it is `true`.

here

Class: Polygon

This class is a member of [nokia.maps.kml](#).

Extends: [nokia.maps.kml.Geometry](#)

Class Summary

This class represents the <Polygon> tag from the KML specification.

[For full details, see [nokia.maps.kml.Polygon](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.kml.Geometry](#) :

[weight](#)

Table 150: Method Summary

Methods
getCenterCoordinate () : {nokia.maps.geo.Coordinate} This method calculates an arithmetic average of all coordinates. This method contains an implementation of nokia.maps.kml.Geometry#getCenterCoordinate.
getPathOuter () : {nokia.maps.geo.Coordinate[]} This method converts the KML property "outerBoundaryIs" to an array of nokia.maps.geo.Coordinate objects.
getStyle () : {Object} This method retrieves the styles for the given object.

Directly Inherited Methods

Inherited from class [nokia.maps.kml.Geometry](#) :

[getCenterCoordinate](#), [getStyle](#)

Inherited from class [nokia.maps.kml.ObservableNode](#) :

[load](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

here

Class Description

This class represents the <Polygon> tag from the KML specification.

Constructor Details

nokia.maps.kml.Polygon(node, polyStyle, balloonStyle)

The constructor initializes a new instance of the class, using the arguments provided by the caller.

Parameters:

node:	{Node}	An XML node from the KML document that is currently being parsed
polyStyle:	{ nokia.maps.kml.PolyStyle }	An object representing the style to use during conversion to a map object
balloonStyle:	{ nokia.maps.kml.BalloonStyle }	An object representing the balloon style to be used during conversion to a map object

Method Details

getCenterCoordinate(): {[nokia.maps.geo.Coordinate](#)}

This method calculates an arithmetic average of all coordinates. This method contains an implementation of [nokia.maps.kml.Geometry#getCenterCoordinate](#).

Returns:

{ nokia.maps.geo.Coordinate }	An instance of <code>nokia.maps.geo.Coordinate</code> containing the averaged coordinates

getPathOuter(): {[nokia.maps.geo.Coordinate](#)[]}

This method converts the KML property "outerBoundaryIs" to an array of [nokia.maps.geo.Coordinate](#) objects.

Returns:

here

```
{nokia.maps.geo.Coordinate[]}
```

Returns an array of *nokia.maps.geo.Coordinate* objects or null if the property "outerBoundaryIs" is absent

getStyle(): {Object}

This method retrieves the styles for the given object. The color and width are not set in the KML file. This method contains an implementation of *nokia.maps.kml.Geometry#getStyle*.

Returns:

{Object} An object with properties "color", "width", "fillColor" and "strokeColor".

Class: Style

This class is a member of *nokia.maps.kml*.

Class Summary

This class represents the <Style> tag from KML specification.

[For full details, see *nokia.maps.kml.Style*]

Table 151: Method Summary

Methods
<i>getStyle</i> () : {<i>nokia.maps.kml.Style</i>} This method gets the style represented by the given object.

Class Description

This class represents the <Style> tag from KML specification.

Constructor Details

nokia.maps.kml.Style(node)

Initialize a new instance of the class with the following parameters.

Parameters:

node: {Node}

here

An XML node from KML document, which is currently parsed.

Method Details

getStyle(): {*nokia.maps.kml.Style*}

This method gets the style represented by the given object..

Returns:

{*nokia.maps.kml.Style*}

A reference to the given instance of the class (this)

Class: StyleContainer

This class is a member of *nokia.maps.kml*.

Class Summary

This class contains an array of *nokia.maps.kml.StyleSelector* objects.

[For full details, see *nokia.maps.kml.StyleContainer*]

Table 152: Method Summary

Methods
getStyleById (id, highlighted) : {<i>nokia.maps.kml.Style</i>} This method gets the concrete <i>nokia.maps.kml.Style</i> for the <i>nokia.maps.kml.StyleSelector</i> matching the id provided by the caller
getStyles () : {<i>nokia.maps.kml.StyleSelector</i>[]} This method gets the internal array of <i>nokia.maps.kml.StyleSelector</i>
push (style) : {Object} This method adds new element to the internal array.

Class Description

This class contains an array of *nokia.maps.kml.StyleSelector* objects. It is mainly used for holding the global styles on which the attribute *id* is set. If a style does not have the *id* attribute, it is not added to the array neither in the constructor nor in the *push()* method.

here

Constructor Details

nokia.maps.kml.StyleContainer(style)

Initialize a new instance of the class with the following parameters.

Parameters:

style: {*nokia.maps.kml.StyleSelector*} [optional]

An instance of *nokia.maps.kml.StyleSelector* that provides the initial style definition; it must contain the attribute *id*; if this argument is not specified, or does not contain *id* attribute, then this initial array is empty

Method Details

getStyleById(id, highlighted): {*nokia.maps.kml.Style*}

This method gets the concrete *nokia.maps.kml.Style* for the *nokia.maps.kml.StyleSelector* matching the *id* provided by the caller

Parameters:

id: {String}

A unique identifier for the *StyleSelector*

highlighted: {Boolean} [optional]

A Boolean value that indicates if the method is to return the style for a highlighted case (*true*); if the value is *true* and if the particular style is not a *StyleMap*, then the method returns *null*, because there is no highlighted style for the *nokia.maps.kml.Style* object

Returns:

{*nokia.maps.kml.Style*}

style matching the *id* specified by the caller

getStyles(): {*nokia.maps.kml.StyleSelector*[]}

This method gets the internal array of *nokia.maps.kml.StyleSelector*

Returns:

here

`{nokia.maps.kml.StyleSelector[]}`

An array of instances of `nokia.maps.kml.StyleSelector`

push(style): {Object}

This method adds new element to the internal array.

Parameters:

style: `{nokia.maps.kml.StyleSelector}`

An instance of `nokia.maps.kml.StyleSelector` that provides the style definition; it must contain the attribute `id`

Returns:

{Object} An array of instances of `nokia.maps.kml.StyleSelectors`

Class: StyleMap

This class is a member of [nokia.maps.kml](#).

Class Summary

This class represents the <StyleMap> element, which provides separate normal and highlighted styles for a placemark so that the highlighted version appears when the user's mouse pointer moves over an icon.

[For full details, see [nokia.maps.kml.StyleMap](#)]

Table 153: Method Summary

Methods
getStyle (highlighted) : <code>{nokia.maps.kml.Style}</code> This method retrieves the style for the given object.

Class Description

This class represents the <StyleMap> element, which provides separate normal and highlighted styles for a placemark so that the highlighted version appears when the user's mouse pointer moves over an icon.

here

Constructor Details

nokia.maps.kml.StyleMap(node, styleContainer)

The constructor initializes a new instance of the class, using the arguments provided by the caller.

Parameters:

node: {Node}

An XML node from the KML document that is being parsed

styleContainer: {[nokia.maps.kml.StyleContainer](#)}

A container with an array of global styles on which the attribute `id` is set to be used by the given `StyleMap`

Method Details

getStyle(highlighted): {[nokia.maps.kml.Style](#)}

This method retrieves the style for the given object.

Parameters:

highlighted: {Boolean} [optional]

A Boolean value that indicates if the method is to return the style for a highlighted case (`true`)

Returns:

{[nokia.maps.kml.Style](#)}

An object encapsulating the style for the given class instance

Class: StyleSelector

This class is a member of [nokia.maps.kml](#).

Class Summary

This class represents the abstract `<StyleSelector>` tag from the KML specification.

[For full details, see [nokia.maps.kml.StyleSelector](#)]

here

Class Description

This class represents the abstract `<StyleSelector>` tag from the KML specification.

Constructor Details

`nokia.maps.kml.StyleSelector`(node)

The constructor initializes a new instance of the class, using the arguments provided by the caller.

Parameters:

node: {Node}

An XML node from the KML document that is being parsed

Class: Vector2

This class is a member of [*nokia.maps.kml*](#).

Class Summary

This class represents the `<kml:vec2>` data type from KML specification.

[For full details, see [*nokia.maps.kml.Vector2*](#)]

Class Description

This class represents the `<kml:vec2>` data type from KML specification.

Constructor Details

`nokia.maps.kml.Vector2`(node)

Initialize a new instance of the class with the following parameters.

Parameters:

node: {Node}

A DOM note from which to extract the attributes "x", "y", "xunits" and "yunits".

Namespace: component

This namespace is a member of [*nokia.maps.kml*](#).

here

Namespace Summary

This namespace defines the KML result set.

Namespace Description

This namespace defines the KML result set.

Class: *KMLResultSet*

This class is a member of [nokia.maps.kml.component](#).

Extends: [nokia.maps.util.OObject](#)

Class Summary

This class represents a result set that can be initiated with a response from a KML file request.

[For full details, see [nokia.maps.kml.component.KMLResultSet](#)]

Table 154: Property Summary

Properties
container : { nokia.maps.map.Container } This property is a container of markers, representing the results.
state : {String} This property holds the state of the container creation.

Table 155: Method Summary

Methods
create () : { nokia.maps.map.Container } This method asynchronously converts kmlDocument provided in the constructor to an instance of nokia.maps.map.Container .
getLineObject (geometry) : { nokia.maps.map.Polyline } This method converts an instance of LinearRing or LineString (a geometry) to a displayable map nokia.maps.map.Polyline .
getPointObject (geometry) : { nokia.maps.map.Marker nokia.maps.map.StandardMarker } This method converts the point geometry to a displayable map object.
getPolygonObject (geometry) : { nokia.maps.map.Polygon } This method converts a polygon geometry to a nokia.maps.map.Polygon that can be displayed on the map.

here

Directly Inherited Methods

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This class represents a result set that can be initiated with a response from a KML file request. If there is an content for particular KML element than [nokia.maps.map.component.InfoBubbles.Bubble](#) is attached to visual representation of this element when user clicks on the element. This can happen only if [nokia.maps.map.component.InfoBubbles](#) component is added to the map:

```
var infoBubbles = new nokia.maps.map.component.InfoBubbles(); //map is an
instance of nokia.maps.map.Display map.components.add(infoBubbles);
```

Constructor Details

nokia.maps.kml.component.KMLResultSet(kmlManager, display)

This method initializes a new instance of the class, using the arguments supplied by the caller.

Parameters:

kmlManager: {[nokia.maps.kml.Manager](#) | [nokia.maps.kml.Document](#)}

An instance of kml.Manager that was used in parsing a KML file (deprecat-
ed: An instance of kml.Document that results from parsing a KML file)

display: {[nokia.maps.map.Display](#)} [optional]

An instance of map.Display so that a map object listener can be added to
ensure support for highlighted styles; if this argument is not provided by the
caller, highlighted styles are not supported

Example:

```
var kmlManager = new nokia.maps.kml.Manager(),
    resultSet,
    container;

kmlManager.addObserver("state", function(kmlManager) {
    if (kmlManager.state == KMLManager.State.FINISHED) {
        resultSet = new nokia.maps.kml.component.KMLResultSet(kmlManager);
        mapDisplay.objects.add(resultSet.create());
    }
}, that);
```

here

```
kmlManager.parseKML("sampleFile.kml");
```

Property Details

container: {[nokia.maps.map.Container](#)}

This property is a container of markers, representing the results.

state: {String}

This property holds the state of the container creation. The possible values are: "initial", "started", "finished".

Method Details

create(): {[nokia.maps.map.Container](#)}

This method asynchronously converts `kmlDocument` provided in the constructor to an instance of [nokia.maps.map.Container](#).

When dealing with heavy KML files, please add the object returned by this method to the map display immediately. This method returns an empty [nokia.maps.map.Container](#) object and starts a [nokia.maps.util.Coroutine](#) which asynchronously fills it with objects held provided by the `kmlDocument` object. As a result, some of the `kmlDocument` objects may be rendered before others are added. When an entire `kmlDocument` is parsed and all the required containers and objects are created, the state of the `KMLResultSet` object is set to "finished". To receive notification of the state change, attach an observer as shown in the example below:

```
var kmlManager = new nokia.maps.kml.Manager(),
    resultSet;

kmlManager.addObserver("state", function (kmlManager) {
    // KML file was successfully loaded
    if (kmlManager.state == "finished") {
        // KML file was successfully parsed
        resultSet = new
nokia.maps.kml.component.KMLResultSet(kmlManager.kmlDocument, map);
        resultSet.addObserver("state", function (resultSet) {
            if (resultSet.state == "finished") {
                // KML object tree was successfully converted into map objects
                // Get the bounding box of container
                boundingBox = resultSet.container.getBoundingBox();
                // Switch the viewport of the map do show all KML map objects within
the container
                if (boundingBox) {
                    map.zoomTo(boundingBox);
                }
            }
        })
    }
})
```

here

```
});
map.objects.add(resultSet.create());
}
});
kmlManager.parseKML("sampleFile.kml");
```

The method uses a modified *Breadth-first search algorithm* to traverse the tree.

Returns:

{*nokia.maps.map.Container*}

Returns map container which will hold map equivalents of KML objects

getLineObject(geometry): {*nokia.maps.map.Polyline*}

This method converts an instance of LinearRing or LineString (a geometry) to a displayable map *nokia.maps.map.Polyline*.

Parameters:

geometry: {*nokia.maps.kml.Point*}

A KML line object or *nokia.maps.kml.LineString* object

Returns:

{*nokia.maps.map.Polyline*}

A displayable map object that can be added directly to an instance of *nokia.maps.map.Display*

getPointObject(geometry): {*nokia.maps.map.Marker* | *nokia.maps.map.StandardMarker*}

This method converts the point geometry to a displayable map object. Depending on the style properties, the resulting object can be an instance of *nokia.maps.map.Marker* or *nokia.maps.map.StandardMarker*

Parameters:

geometry: {*nokia.maps.kml.Point*}

A KML point object

here

Returns:

```
{nokia.maps.map.Marker
|
nokia.maps.map.StandardMarker}
```

An object that can be displayed on the map by adding it to an instance of *nokia.maps.map.Display*

getPolygonObject(geometry): {*nokia.maps.map.Polygon*}

This method converts a polygon geometry to a *nokia.maps.map.Polygon* that can be displayed on the map.

Parameters:

geometry: {*nokia.maps.kml.Polygon*}

A KML polygon object

Returns:

```
{nokia.maps.map.Polygon}
```

A displayable polygon that can be added directly to an instance of *nokia.maps.map.Display*

Namespace: map

This namespace is a member of *nokia.maps*.

Namespace Summary

This namespace defines classes that display the map and support management of map objects such as markers, geometric shapes, etc.

Namespace Description

This namespace defines classes that display the map and support management of map objects such as markers, geometric shapes, etc.

here

Class: Circle

This class is a member of [nokia.maps.map](#).

Class Summary

This class defines a map object with a circular shape.

[For full details, see [nokia.maps.map.Circle](#)]

Table 156: Property Summary

Properties
center : { nokia.maps.geo.Coordinate } An object holding the coordinates of the center of the circle.
precision : {Number} This property holds the precision of the circle as a number of segments to be used when drawing the circle.
radius : {Number} The radius of the circle in meters.

Directly Inherited Properties

Inherited from class [nokia.maps.map.Polygon](#) :

[brush](#)

Inherited from class [nokia.maps.map.Polyline](#) :

[arrows](#), [path](#), [pen](#)

Inherited from class [nokia.maps.map.Spatial](#) :

[simplify](#)

Inherited from class [nokia.maps.map.Object](#) :

[CHANGE_SPATIAL](#), [CHANGE_VISUAL](#), [CHANGE_ZINDEX](#), [visibility](#), [zIndex](#)

Inherited from class [nokia.maps.dom.EventTarget](#) :

[isDraggable](#), [isEventTarget](#), [parentNode](#), [parentNodes](#)

Inherited from class [nokia.maps.map.provider.IData](#) :

[id](#)

here

Method Summary

Directly Inherited Methods

Inherited from class *nokia.maps.map.Polyline* :

getNearest, getNearestIndex

Inherited from class *nokia.maps.map.Object* :

destroy, getBoundingBox, getDisplay, getParent, getProvider, getRoot, isVisible

Inherited from class *nokia.maps.util.OObject* :

addObserver, get, remove, removeObserver, set

Inherited from class *nokia.maps.dom.EventTarget* :

addListener, addListenerNS, addListeners, disableDrag, disableUserSelect, dispatch, enableDrag, enableUserSelect, hitTest, insertListener, insertListenerNS, removeAllListeners, removeListener, removeListenerNS

Inherited from class *nokia.maps.map.provider.IData* :

getInvalidations

Event Summary

Directly Inherited Events

Inherited from class *nokia.maps.dom.MouseEventTarget* :

click, dblclick, longpress, mousedown, mouseenter, mouseleave, mousemove, mouseout, mouseover, mouseup, mousewheel

Inherited from class *nokia.maps.dom.DragEventTarget* :

drag, dragend, dragenter, dragleave, dragover, dragstart, drop

Inherited from class *nokia.maps.dom.TouchEventTarget* :

dbltap, gesturechange, gestureend, gesturestart, longpress, tap, touchend, touchmove, touchstart

Class Description

This class defines a map object with a circular shape.

Constructor Details

nokia.maps.map.Circle(center, radius, props)

here

This method initializes an instance of `nokia.maps.map.Circle`.

Parameters:

center:	<code>{nokia.maps.geo.Coordinate}</code>	An object providing the geographical coordinates of the center of the circle
radius:	<code>{Number}</code>	A value that provides the length of the radius of the circle in meters
props:	<code>{nokia.maps.map.Circle.Properties}</code>	An object that specifies object properties and their initial values (among these, <code>precision</code> has a significant impact on the shape of the circle - please see nokia.maps.map.Circle.Properties)

Property Details

center: `{nokia.maps.geo.Coordinate}`

An object holding the coordinates of the center of the circle.

precision: `{Number}`

This property holds the precision of the circle as a number of segments to be used when drawing the circle. The value must be in the range between [4 ... 360], where 60 is the default. Note that the lower the value the more angular and the less circle-like the shape appears and, conversely, the higher the value the smoother and more rounded the result. Thus, starting at the extreme low end of the possible values, 4 produces a square, 6 a hexagon, while 30 results in a circle-like shape, although it appears increasingly angular as the zoom level increases (as you zoom in), and finally 360 produces a smooth circle.

Default Value: 60

radius: `{Number}`

The radius of the circle in meters. Negative values are not allowed.

Interface: Properties

This interface is a member of [nokia.maps.map.Circle](#).

here

Interface Summary

This interface defines the properties (keys) that can be passed to the constructor of `nokia.maps.map.Circle`.

[For full details, see `nokia.maps.map.Circle.Properties`]

Table 157: Property Summary

Properties
<code>precision</code> : {Number}
This property provides the value for <code>nokia.maps.map.Circle#precision</code> .

Directly Inherited Properties

Inherited from class `nokia.maps.map.Polygon.Properties` :

`brush`

Inherited from class `nokia.maps.map.Polyline.Properties` :

`arrows`, `pen`

Inherited from class `nokia.maps.map.Spatial.Properties` :

`simplify`

Inherited from class `nokia.maps.map.Object.Properties` :

`visibility`, `zIndex`

Interface Description

This interface defines the properties (keys) that can be passed to the constructor of `nokia.maps.map.Circle`.

Property Details

`precision`: {Number}

This property provides the value for `nokia.maps.map.Circle#precision`.

Class: Container

This class is a member of `nokia.maps.map`.

Extends: `nokia.maps.map.Object`

here

Class Summary

This class defines a container for objects that can be displayed on the map.

[For full details, see [nokia.maps.map.Container](#)]

Table 158: Property Summary

Properties
objects : { nokia.maps.util.OList }
This property holds the objects that have been added to the given container.

Directly Inherited Properties

Inherited from class [nokia.maps.map.Object](#) :

[CHANGE_SPATIAL](#), [CHANGE_VISUAL](#), [CHANGE_ZINDEX](#), [visibility](#), [zIndex](#)

Inherited from class [nokia.maps.dom.EventTarget](#) :

[isDraggable](#), [isEventTarget](#), [parentNode](#), [parentNodes](#)

Inherited from class [nokia.maps.map.provider.IData](#) :

[id](#)

Table 159: Method Summary

Methods
contains (object) : {Boolean}
This method checks if the container contains the map object provided by the caller.
getBoundingBox () : { nokia.maps.geo.BoundingBox }
This method calculates the outer bounding box of the given container.

Directly Inherited Methods

Inherited from class [nokia.maps.map.Object](#) :

[destroy](#), [getBoundingBox](#), [getDisplay](#), [getParent](#), [getProvider](#), [getRoot](#), [isVisible](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Inherited from class [nokia.maps.dom.EventTarget](#) :

here

addListener, addListenerNS, addListeners, disableDrag, disableUserSelect, dispatch, enableDrag, enableUserSelect, hitTest, insertListener, insertListenerNS, removeAllListeners, removeListener, removeListenerNS

Inherited from class *nokia.maps.map.provider.IData* :

getInvalidations

Event Summary

Directly Inherited Events

Inherited from class *nokia.maps.dom.MouseEventTarget* :

click, dblclick, longpress, mousedown, mouseenter, mouseleave, mousemove, mouseout, mouseover, mouseup, mousewheel

Inherited from class *nokia.maps.dom.DragEventTarget* :

drag, dragend, dragenter, dragleave, dragover, dragstart, drop

Inherited from class *nokia.maps.dom.TouchEventTarget* :

dbltap, gesturechange, gestureend, gesturestart, longpress, tap, touchend, touchmove, touchstart

Class Description

The class Container groups objects (instances of *nokia.maps.map.Object*), allowing operations to be performed on them as a unit. For example, you can set the visibility of the container and thus show or hide all the objects it includes, without having to change the visibility of each object individually. An instance of Container can include other containers as well as objects such as markers, polylines, polygons, etc.

Constructor Details

nokia.maps.map.Container(props)

This method creates an instance of Container. The caller provides the object to be added to the container, as well as container initialization properties.

Parameters:

props: {*nokia.maps.map.Container.Properties*}

The initialization properties for the container

here

Property Details

objects: {[*nokia.maps.util.OLList*](#)}

This property holds the objects that have been added to the given container. The property can be managed, using methods defined on `nokia.maps.util.OLList`.

Method Details

contains(object): {Boolean}

This method checks if the container contains the map object provided by the caller.

Parameters:

object: {[*nokia.maps.map.Object*](#)}

An object whose presence in the given instance of Container is to be verified.

Since: 2.5

Returns:

{Boolean} true if the container contains the object or they are identical, otherwise false.

getBoundingBox(): {[*nokia.maps.geo.BoundingBox*](#)}

This method calculates the outer bounding box of the given container. This means a bounding box encompassing all the map objects in the container, including all child containers. If the container does not contain any objects or child containers, the method returns `null`. If the container contains only one object that has no geographical dimensions, for example an instance of [*nokia.maps.map.Marker*](#) or [*nokia.maps.map.StandardMarker*](#), then the returned bounding box may have the size of zero.

Returns:

{[*nokia.maps.geo.BoundingBox*](#)}

The calculated geographic outer bounding box of the given container or `null`, if the container has no geographic dimensions.

here

Interface: Properties

This interface is a member of [nokia.maps.map.Container](#).

Interface Summary

This interface defines the properties (keys) that can be passed to the constructor of [nokia.maps.map.Container](#).

[For full details, see [nokia.maps.map.Container.Properties](#)]

Table 160: Property Summary

Properties
objects: { nokia.maps.util.OList } This property holds the initial set of map objects that have been added to the container.

Directly Inherited Properties

Inherited from class [nokia.maps.map.Object.Properties](#) :

[visibility](#), [zIndex](#)

Interface Description

This interface defines the properties (keys) that can be passed to the constructor of [nokia.maps.map.Container](#).

Property Details

objects: {[nokia.maps.util.OList](#)}

This property holds the initial set of map objects that have been added to the container. The property can be managed, using methods defined on [nokia.maps.util.OList](#).

Class: Display

This class is a member of [nokia.maps.map](#).

Extends: [nokia.maps.map.Container](#)

Class Summary

This class displays a map and map objects such as markers, polylines, polygons, etc.

here

[For full details, see [nokia.maps.map.Display](#)]

Table 161: Property Summary

Properties
animation: {nokia.maps.util.OList} This property specifies the currently supported map animation types.
availableBaseMapTypes: {nokia.maps.util.OList} This property holds a list of all available nokia.maps.map.provider.Providers for base map types.
availableOverlays: {nokia.maps.util.OList} This property holds a list of all available nokia.maps.map.provider.Providers for map overlays.
baseMapType: {nokia.maps.map.provider.Provider} This property holds the current base map type to be used for display (such as satellite, terrain, hybrid).
bounceEnabled: {Boolean} This property keeps a flag to signal if the bouncing effect is activated or not
center: {nokia.maps.geo.Coordinate} This property contains the coordinates of the map center.
components: {nokia.maps.util.OList} This property holds a list of all components that have been added to the map (to the given instance of nokia.maps.map.Display).
copyrightAlignment: {String} This property specifies the position of the copyright information relative to the display.
readonly copyrightHeight: {Number} This property holds the height in pixels of the frame containing the text "Copyright by .
readonly copyrightPosition: {nokia.maps.util.IPoint} This property holds the position of the frame containing the label "Copyright by ".
readonly copyrightWidth: {Number} This property holds the width in pixels of the frame containing the text "Copyright by .
default2dRenderMode: {String} This property holds the type of the render mode for 2d map.
fixedCenter: {Boolean} This property is a flag indicating if the center of the map should remain unchanged if the display is resized or padding changes.
heading: {Number} This property holds the heading (bearing) of the map in degrees.

here

Properties

readonly **height**: {Number}

This property holds the height of the map area in pixels.

static **InteractionModifiers**:

This variable defines a set of bitmask values that can be passed to the method *nokia.maps.map.Display.beginInteraction*.

margin: {Number}

This property holds the size of the supplemental rendering area.

maxHeading: {Number}

This property holds the maximum supported map heading (bearing).

maxTilt: {Number}

This property holds the maximum supported map tilt.

maxZoomLevel: {Number}

This property holds the maximum zoom level supported by the map (considering the current heading/tilting).

minHeading: {Number}

This property holds the minimum supported map heading (bearing).

minTilt: {Number}

This property holds the minimum supported map tilt.

minZoomLevel: {Number}

This property holds the minimum zoom level supported by the map (considering the current heading/tilting).

static **NORMAL**: {*nokia.maps.map.provider.TileProvider*}

This type indicates a normal street map.

onError:

Function that is invoked when error condition occurs during baseMapType switching (for example no StreetLevel coverage exists for the given coordinate).

overlays: {*nokia.maps.util.OLList*}

This property holds the list of all map overlays which are currently active for the map.

padding: {*nokia.maps.util.IBox*}

This property defines the padding in pixels for each side of the map display, thus setting a virtual viewport.

readonly **poweredByHeight**: {Number}

This property holds the height in pixels of the frame containing the logo "Powered by .

readonly **poweredByPosition**: {*nokia.maps.util.IPoint*}

This property holds the position of the frame containing the label "Powered by .

readonly **poweredByWidth**: {Number}

This property holds the width in pixels of the frame containing the logo "Powered by .

here

Properties

static **SATELLITE**: {[nokia.maps.map.provider.TileProvider](#)}

This type indicates a hybrid satellite map with street information on top.

static **SATELLITE_PLAIN**: {[nokia.maps.map.provider.TileProvider](#)}

This type indicates a satellite map.

static **SMART_PT**: {[nokia.maps.map.provider.TileProvider](#)}

This type indicates a public transport map.

static **SMARTMAP**: {[nokia.maps.map.provider.TileProvider](#)}

This type indicates a smart map.

static **TERRAIN**: {[nokia.maps.map.provider.TileProvider](#)}

This type indicates a topographical map, where shading and color convey the shape of the terrain.

tilt: {Number}

This property holds the value of the map tilt in degrees.

static **TRAFFIC**: {[nokia.maps.map.provider.TileProvider](#)}

This type indicates a traffic info map.

static **TRAFFIC_INCIDENTS**: {[nokia.maps.map.provider.TileProvider](#)}

This type indicates a map showing the location of traffic incidents.

readonly **width**: {Number}

This property holds the width of the map area in pixels.

zoomLevel: {Number}

This property holds the zoom level of the current view.

Directly Inherited Properties

Inherited from class [nokia.maps.map.Container](#) :

[objects](#)

Inherited from class [nokia.maps.map.Object](#) :

[CHANGE_SPATIAL](#), [CHANGE_VISUAL](#), [CHANGE_ZINDEX](#), [visibility](#), [zIndex](#)

Inherited from class [nokia.maps.dom.EventTarget](#) :

[isDraggable](#), [isEventTarget](#), [parentNode](#), [parentNodes](#)

Inherited from class [nokia.maps.map.provider.IData](#) :

[id](#)

here

Table 162: Method Summary

Methods
<code>addComponent</code> (component) : { <i>nokia.maps.map.component.Component</i> } This method adds a new component to the given Display instance.
static <code>beginControl</code> (mode, kinetics, x, y) This method signals the begin of a control operation.
static <code>beginInteraction</code> (modifiers, kinetics, timeStamp, x, y, biX, biY) This method signals the beginning of an interaction.
<code>blur</code> () Removes keyboard focus from the display.
<code>bounce</code> (direction, atX, atY) Method is responsible for map bouncing effect if one is supported with the current baseMapType.
<code>capture</code> (left, top, right, bottom) : { <i>nokia.maps.gfx.Graphics</i> } This method requests an image capturing of the viewport at the given rectangle relative to the viewport's origin.
static <code>control</code> (timeStamp, moveX, moveY, moveZ, angleX, angleY, angleZ, zoom) This method process a control operation.
<code>destroy</code> () This method destroys the map Display instance and frees the resources.
static <code>endControl</code> (adjust) This method signals the end of a control operation.
static <code>endInteraction</code> () This method signals the end of an interaction.
<code>focus</code> () Gives keyboard focus to the display.
<code>geoToPixel</code> (coord) : { <i>nokia.maps.util.Point</i> } This method translates the received point object containing WGS84 coordinates into pixel coordinates relative to the top left corner of the map view.
<code>getBestZoomLevel</code> (bBoxes) : {Number} This method computes the minimum zoom level at which each of the given bounding boxes fit into the display area.
<code>getBoundingBox</code> () : { <i>nokia.maps.geo.BoundingBox</i> } This method calculates the outer bounding box of all map objects in the given Display instance, including all child containers.
<code>getCam</code> () : { <i>nokia.maps.map.ICam</i> } This method returns the map camera

Methods

getComponentById (id) : {*nokia.maps.map.component.Component*}

This method returns the first component with the given identifier or null if no component with such an identifier is currently attached to the Display instance.

getObjectAt (x, y) : {*nokia.maps.map.Object* | undefined}

A method to obtain the topmost visible object at the given position relative to the viewport's origin.

getObjectsAt (x, y) : {*nokia.maps.map.Object*[]}}

A method to obtain all objects that are visible at the given pixel position relative to the viewport's origin.

getObjectsWithin (left, top, right, bottom) : {*nokia.maps.map.Object*[]}}

This method returns all objects found within a rectangle defined in terms of the pixel coordinates of its top left and bottom right corners.

getView () : {*nokia.maps.map.IView*}

This method returns the map view

getViewBounds () : {*nokia.maps.geo.BoundingBox*}

This method retrieves the outer bounding box of the map view (the smallest bounding box covering all visible points).

getZoom (x, y) : {Number}

This method returns zoom level at provided screen position.

static **interact** (timeStamp, x, y, biX, biY)

This method processes an interaction based on the screen coordinates of the viewport.

pan (startX, startY, endX, endY, animation)

This method pans the map from the start point to the end point specified by the caller.

pixelToGeo (x, y) : {*nokia.maps.geo.Coordinate*}

This method translates a pixel position within the viewport to geo coordinates.

removeComponent (component) : {Number}

This method decrements the reference counter of the supplied component and if the counter has reached zero, the method removes the component from display.

setAttributes (animation, center, level, tilt, heading)

This method sets a number of properties of the map.

setBaseMapType (provider, animate, view)

This method sets base map type provider and optional transition between old and new provider can be involved.

setCam (cam, animation) : {*nokia.maps.map.ICam*}

This method sets the new camera for the map.

setCenter (coord, animation)

This method centers the map on the location specified by the caller.

setCopyrightAlignment (alignment)

here

Methods

This method sets the position of the copyright information.

setHeading (heading, animation)

This method sets the heading (bearing) of the map.

setPadding (padding1, padding2, padding3, padding4)

This method sets the property *nokia.maps.map.Display#padding*.

setTilt (tilt, animation)

This method sets the tilt of the map.

setView (view, animation) : {*nokia.maps.map.IView*}

This method sets the new view for the map.

setZoomLevel (level, animation, toX, toY)

This method sets the zoom level to the value specified by the caller.

update (delay, quick) : {*nokia.maps.map.Display*}

This method causes the current map view to be re-rendered.

zoomTo (boundingBox, keepCenter, animation)

This method zooms the map to ensure that the bounding box provided by the caller is visible in its entirety in the map viewport.

Directly Inherited Methods

Inherited from class *nokia.maps.map.Container* :

contains, *getBoundingBox*

Inherited from class *nokia.maps.map.Object* :

destroy, *getBoundingBox*, *getDisplay*, *getParent*, *getProvider*, *getRoot*, *isVisible*

Inherited from class *nokia.maps.util.OObject* :

addObserver, *get*, *remove*, *removeObserver*, *set*

Inherited from class *nokia.maps.dom.EventTarget* :

addListener, *addListenerNS*, *addListeners*, *disableDrag*, *disableUserSelect*, *dispatch*, *enableDrag*, *enableUserSelect*, *hitTest*, *insertListener*, *insertListenerNS*, *removeAllListeners*, *removeListener*, *removeListenerNS*

Inherited from class *nokia.maps.map.provider.IData* :

getInvalidations

here

Table 163: Event Summary

Events
basemapchangeend This event is fired to mark the end of a change from one base map type to another (for example NORMAL to SATELLITE).
basemapchangestart This event is fired to mark the start of a change from one base map type to another (for example NORMAL to SATELLITE).
displayready This event is fired when the map has been initialized and is ready to support interactive events, transitions and animations.
mapvalueexceeded This event is fired if a received value exceeds the permitted range.
mapviewchange This event is fired each time after the map has been rendered in response to a mapviewchange event (see nokia.maps.map.Display.MapViewChangeEvent).
mapviewchangeend This event is fired when the map display has been rendered and all map view changes have been processed (see also nokia.maps.map.Display.MapViewChangeEvent).
mapviewchangestart This event is fired immediately after the map view has been changed (see nokia.maps.map.Display.MapViewChangeEvent).
resize This event is fired each time a change of the size of the display is detected.
resizeend This event is fired when a resizing operation has finished (when no more changes to the size of the display have been detected for a certain amount of time).
resizestart This event is fired immediately after the process of resizing the display has begun and then periodically until the process has completed, when resizeend is fired.
transitionend This event is fired immediately after a map animation has finished or a transition from one base map type to another (in the latter case the event precedes nokia.maps.map.Display#basemapchangeend).
transitionstart This event is fired immediately after a map animation starts or a transition from one base map type to another (in the latter case the event follows nokia.maps.map.Display#basemapchangestart).

Directly Inherited Events

Inherited from class [nokia.maps.dom.MouseEventTarget](#) :

here

click, dblclick, longpress, mousedown, mouseenter, mouseleave, mousemove, mouseout, mouseover, mouseup, mousewheel

Inherited from class *nokia.maps.dom.DragEventTarget* :

drag, dragend, dragenter, dragleave, dragover, dragstart, drop

Inherited from class *nokia.maps.dom.TouchEventTarget* :

dbltap, gesturechange, gestureend, gesturestart, longpress, tap, touchend, touchmove, touchstart

Class Description

This class displays a map, allowing multiple optional layers (containers) and objects such as markers, polylines, polygons, circles, etc., to be added to it.

Constructor Details

nokia.maps.map.Display(container, props)

This method displays a map, allowing multiple containers and objects to be added to it.

Parameters:

container:	{HTML <code>E</code> lement}
	The DOM node where the map should be displayed
props:	{ <i>nokia.maps.map.Display.Properties</i> } [optional]
	An object containing initialization properties

Property Details

animation: {*nokia.maps.util.OList*}

This property specifies the currently supported map animation types.

Animations depend on the platform and the rendering engine used, therefore this property must be checked at run-time to see if a specific animation is supported by the current rendering engine on the current platform.

In all cases, the following values for this property are supported:

- "none" - no animation
- "default" - default animation

here

Note 1: The default animation may be set to "none", which means that the rendering engine on the current platform does not support animation.

Note 2: If you specify an incorrect animation name, the animation type automatically reverts to "default".

availableBaseMapTypes: {[nokia.maps.util.OList](#)}

This property holds a list of all available [nokia.maps.map.provider.Providers](#) for base map types.

availableOverlays: {[nokia.maps.util.OList](#)}

This property holds a list of all available [nokia.maps.map.provider.Providers](#) for map overlays.

baseMapType: {[nokia.maps.map.provider.Provider](#)}

This property holds the current base map type to be used for display (such as satellite, terrain, hybrid). The given type must be one of the supported types. See [nokia.maps.map.Display#availableBaseMapTypes](#)

bounceEnabled: {Boolean}

This property keeps a flag to signal if the bouncing effect is activated or not

Default Value: true

center: {[nokia.maps.geo.Coordinate](#)}

This property contains the coordinates of the map center.

components: {[nokia.maps.util.OList](#)}

This property holds a list of all components that have been added to the map (to the given instance of `nokia.maps.map.Display`).

copyrightAlignment: {String}

This property specifies the position of the copyright information relative to the display. The allowed values are:

here

- topleft
- topcenter
- topright
- middleleft
- middlecenter
- middleright
- bottomleft
- bottomcenter
- bottomright

Default Value: "bottomleft"

readonly **copyrightHeight**: {Number}

This property holds the height in pixels of the frame containing the text "Copyright by ...".

readonly **copyrightPosition**: {*nokia.maps.util.IPoint*}

This property holds the position of the frame containing the label "Copyright by ".

readonly **copyrightWidth**: {Number}

This property holds the width in pixels of the frame containing the text "Copyright by ...".

default2dRenderMode: {String}

This property holds the type of the render mode for 2d map. Possible values are:

- "dom"
- "canvas"

If set to "canvas" but environment doesn't support "canvas" element "dom" rendering will be used

fixedCenter: {Boolean}

This property is a flag indicating if the center of the map should remain unchanged if the display is resized or padding changes.

here

Default Value: `true`

heading: {Number}

This property holds the heading (bearing) of the map in degrees. The property depends on the platform used and is only available if `maxHeading` is greater than 0.

readonly **height:** {Number}

This property holds the height of the map area in pixels.

static **InteractionModifiers:**

This variable defines a set of bitmask values that can be passed to the method [nokia.maps.map.Display.beginInteraction](#).

margin: {Number}

This property holds the size of the supplemental rendering area. The renderer must be prepared to allow rapid mouse panning within the displayed area extended on each side by the given margin (in pixels).

Default Value: `64`

maxHeading: {Number}

This property holds the maximum supported map heading (bearing). You can obtain the maximum supported heading by invoking the method `get()` on an instance of `Display`, with the name of this property as the only argument. For example, if "myMap" is an instance of `map.Display`, then the code `myMap.get("maxHeading")` obtains the current value of `maxHeading`.

maxTilt: {Number}

This property holds the maximum supported map tilt. You can obtain the maximum supported tilt by invoking the method `get()` on an instance of `Display`, with the name of this property as the only argument. For example, if "myMap" is an instance of `map.Display`, then the code `myMap.get("maxTilt")` obtains the current value of `maxTilt`.

here

maxZoomLevel: {Number}

This property holds the maximum zoom level supported by the map (considering the current heading/tilting). You can obtain the maximum supported zoom level by invoking the method `get()` on an instance of `Display`, with the name of this property as the only argument. For example, if "myMap" is an instance of `map.Display`, then the code `myMap.get("maxZoomLevel")` obtains the current value of `maxZoomLevel`.

minHeading: {Number}

This property holds the minimum supported map heading (bearing). You can obtain the minimum supported heading by invoking the method `get()` on an instance of `Display`, with the name of this property as the only argument. For example, if "myMap" is an instance of `map.Display`, then the code `myMap.get("minHeading")` obtains the current value of `minHeading`.

minTilt: {Number}

This property holds the minimum supported map tilt. You can obtain the minimum supported tilt by invoking the method `get()` on an instance of `Display`, with the name of this property as the only argument. For example, if "myMap" is an instance of `map.Display`, then the code `myMap.get("minTilt")` obtains the current value of `minTilt`.

minZoomLevel: {Number}

This property holds the minimum zoom level supported by the map (considering the current heading/tilting). You can obtain the minimum supported zoom level by invoking the method `get()` on an instance of `Display`, with the name of this property as the only argument. For example, if "myMap" is an instance of `map.Display`, then the code `myMap.get("minZoomLevel")` obtains the current value of `minZoomLevel`.

static NORMAL: {*nokia.maps.map.provider.TileProvider*}

This type indicates a normal street map.

onError:

here

Function that is invoked when error condition occurs during `baseMapType` switching (for example no `StreetLevel` coverage exists for the given coordinate). By default does nothing.

overlays: `{nokia.maps.util.OLList}`

This property holds the list of all map overlays which are currently active for the map.

padding: `{nokia.maps.util.IBox}`

This property defines the padding in pixels for each side of the map display, thus setting a virtual viewport. The virtual viewport does not affect the placing or the size of the visible map in the DOM node, but defines an area which excludes strips along the edges of the map that contain the UI elements zoom bar, copyright and map-type selector (if these elements are displayed). The virtual viewport is taken into consideration when setting the map center, when obtaining the map view bounds and when zooming to a bounding box. Its other purpose is to help prevent objects from overlapping very close to the edges of the visible map.

The map viewport determined by padding is taken into account when setting the map center, which is the center of the virtual viewport, not of the visible map. Furthermore, the viewport is used by the following methods:

- [nokia.maps.map.Display#setCenter](#) - the map is centered in the virtual viewport, not in the visible area of the map
- [nokia.maps.map.Display#getViewBounds](#) - the method calculates and returns the bounding box of the virtual view port
- [nokia.maps.map.Display#zoomTo](#) - the method sets the zoom level to ensure that the entire bounding box passed to it is visible in the virtual viewport

padding cannot be changed directly. Please use the methods [nokia.maps.map.Display#setPadding](#) or the inherited [nokia.maps.util.OObject#set](#) instead. For example, you can call the method `set ( )` as follows:

```
set("padding", aPadding);
```

readonly **poweredByHeight:** `{Number}`

This property holds the height in pixels of the frame containing the logo "Powered by ...".

readonly **poweredByPosition:** `{nokia.maps.util.IPoint}`

here

This property holds the position of the frame containing the label "Powered by ".

readonly **poweredByWidth**: {Number}

This property holds the width in pixels of the frame containing the logo "Powered by ...".

static **SATELLITE**: {*nokia.maps.map.provider.TileProvider*}

This type indicates a hybrid satellite map with street information on top.

static **SATELLITE_PLAIN**: {*nokia.maps.map.provider.TileProvider*}

This type indicates a satellite map.

static **SMART_PT**: {*nokia.maps.map.provider.TileProvider*}

This type indicates a public transport map. It uses the smart map color scheme with additional public transport route lines on top. Availability of public transport information may vary.

static **SMARTMAP**: {*nokia.maps.map.provider.TileProvider*}

This type indicates a smart map. It is similar to normal street map, but uses an optimized color set ready to show additional overlay information.

static **TERRAIN**: {*nokia.maps.map.provider.TileProvider*}

This type indicates a topographical map, where shading and color convey the shape of the terrain.

tilt: {Number}

This property holds the value of the map tilt in degrees. The property is platform dependent and is available only if `maxTilt` is greater than 0.

static **TRAFFIC**: {*nokia.maps.map.provider.TileProvider*}

This type indicates a traffic info map. Since version 2.5.3 it can be used as a base map. Availability of traffic information may vary.

here

static **TRAFFIC_INCIDENTS**: {[nokia.maps.map.provider.TileProvider](#)}

This type indicates a map showing the location of traffic incidents. It is an overlay which can be added on top of other base map types. Availability of traffic information may vary.

readonly **width**: {Number}

This property holds the width of the map area in pixels.

zoomLevel: {Number}

This property holds the zoom level of the current view.

Method Details

addComponent(component): {[nokia.maps.map.component.Component](#)}

This method adds a new component to the given Display instance. If a component with the same id already exists (has been added previously), the method returns that component after incrementing its reference counter (the method does not create/add multiple instances of the same component).

Parameters:

component: {[nokia.maps.map.component.Component](#)}

A reference to the component to be added

Returns:

{[nokia.maps.map.component.Component](#)}

A reference to the newly added component or to the existing component with the same identifier

Example:

```
var component = display.addComponent(new
nokia.maps.map.component.zoom.DoubleClick());
```

static **beginControl**(mode, kinetics, x, y)

here

This method signals the begin of a control operation. Any control operation currently in progress is stopped without a kinetic effect. A current control operation is also stopped if any of the methods `setView()`, `setCam()` or `beginInteraction()` are called.

Parameters:

mode: {String}

A constant which must be either ("cam") or the view ("view") to indicate if the camera of view is to be controlled.

kinetics: {nokia.maps.map.IKinetics}

A value indicating the kinetic effect to use when the control ends. If null, no kinetic effect occurs.

x: {Number} [optional]

The X coordinate of the screen location, where a view control operation is applied; if NaN the screen center is used.

y: {Number} [optional]

The Y coordinate of the screen location, where a view control operation is applied; if NaN the screen center is taken instead.

static **beginInteraction**(modifiers, kinetics, timeStamp, x, y, biX, biY)

This method signals the beginning of an interaction. Any interaction already in progress is stopped, without a kinetic effect. A current interaction is also stopped if `setView()`, `setCam()` or `beginControl()` are called.

Parameters:

modifiers: {Number}

A bit mask of all enabled modification types of the map view (See [nokia.maps.map.Display.InteractionModifiers](#)). Apart from the COORD modifier, all other modifiers are evaluated in descending order of their numeric values. Only the first two which are set are taken into consideration.

kinetics: {nokia.maps.map.IKinetics}

here

A value specifying the kinetics to use when the interaction ends. If the argument is `null`, no kinetic effect is executed.

`timeStamp:` {Number}

The time stamp of interaction, needed for the computation of the kinetic effect.

`x:` {Number}

The starting X coordinate of the interaction.

`y:` {Number}

The starting Y coordinate of the interaction.

`biX:` {Number} [optional]

The second starting X coordinate of the interaction, only needed for bifocal interaction.

`biY:` {Number} [optional]

The second starting Y coordinate of the interaction, only needed for bifocal interaction.

blur()

Removes keyboard focus from the display.

Since: 2.5

bounce(direction, atX, atY)

Method is responsible for map bouncing effect if one is supported with the current `baseMapType`.

Parameters:

`direction:` {Boolean} [optional]

if value is true map bounces in opposite direction

`atX:` {Number} [optional]

here

screen X coordinate at which bounce effect takes place (defaults to screen center)

atY: {Number} [optional]

screen Y coordinate at which bounce effect takes place (defaults to screen center)

capture(left, top, right, bottom): {*nokia.maps.gfx.Graphics*}

This method requests an image capturing of the viewport at the given rectangle relative to the viewport's origin.

Parameters:

left: {Number} [optional, default: 0]
The X coordinate of the left edge of the rectangle.

top: {Number} [optional, default: 0]
The Y coordinate of the top edge of the rectangle

right: {Number} [optional, default: width]
The X coordinate of the right edge of the rectangle

bottom: {Number} [optional, default: height]
The Y coordinate of the bottom edge of the rectangle

Returns:

{*nokia.maps.gfx.Graphics*}

static **control**(timeStamp, moveX, moveY, moveZ, angleX, angleY, angleZ, zoom)

This method process a control operation. If no control operation is currently in progress (beginControl() has not been called or an endControl() has been called), the call to this method is ignored.

The x/y/z axes for movement and rotation are dependent on the control mode:

here

- In "cam" mode, the axes are relative to the current orientation of the cam.
- In "view" mode, the axes are relative to the current heading of the view.

Parameters:

timeStamp: {Number}

A timestamp value indicating when control call occurred, needed for computation of the kinetic effect.

moveX: {Number} [optional, default: 0]

A value indicating the move of the cam or view along the screen x axis as levels per millisecond.

moveY: {Number} [optional, default: 0]

A value indicating the move of the cam or view along the screen x axis as levels per millisecond.

moveZ: {Number} [optional, default: 0]

A value indicating the move of the cam or view along the screen z axis as levels per millisecond.

angleX: {Number} [optional, default: 0]

A value indicating the rotation of the cam or view about the screen x axis as degrees per millisecond.

angleY: {Number} [optional, default: 0]

A value indicating the rotation of the cam or view about the screen x axis as degrees per millisecond.

angleZ: {Number} [optional, default: 0]

A value indicating the rotation of the cam or view about the screen x axis as degrees per millisecond.

zoom: {Number} [optional, default: 0]

A value indicating the zoom of the view as zoom levels per millisecond; ignored in "cam" mode.

here

destroy()

This method destroys the map `Display` instance and frees the resources. The method may take up to a minute to complete, but afterwards, all of `Display`'s properties are set to `null` and its methods are bound to an empty function.

static endControl(adjust)

This method signals the end of a control operation. If `beginControl()` has received the `kinetics` argument, the kinetic effect is performed. If no control operation is currently in progress (`beginControl()` has not been called or `endControl()` has been invoked), the call to `endControl()` is ignored.

Parameters:

adjust: {Function} [optional]

An optional callback function to adjust the end-view or -cam (inclusive the optional kinetic effect). For example, it could be useful that a minimal change between start- and end-state is assured (e.g. at least 1 zoom level change) or to adjust the final state to an interval (e.g. 90°)

The function must be able to receive the following arguments:

- {`nokia.maps.map.ICam`|`slocation.map.IView`} `beginState` The engine's cam or view state when `beginControl()` was invoked
- {`nokia.maps.map.ICam`|`slocation.map.IView`} `endState` The engine's cam or view state after the control operation has completed (inclusive the kinetic effect)

The function must return the adjusted `endState` as [`nokia.maps.map.ICam`](#) or [`nokia.maps.map.IView`](#), or `false` to terminate the control operation at once without a kinetic effect.

static endInteraction()

This method signals the end of an interaction. If the argument `kinetics` was passed to the previous call to `beginInteraction()`, the kinetic effect is performed. If no interaction is currently in progress, (`beginInteraction()` has not been triggered, or if `endInteraction()` has been called), the call to `endInteraction()` is ignored.

here

focus()

Gives keyboard focus to the display.

Since: 2.5

geoToPixel(coord): {*nokia.maps.util.Point*}

This method translates the received point object containing WGS84 coordinates into pixel coordinates relative to the top left corner of the map view. The coordinates of the top left corner of the map are (0,0).

Note that the result values outside the visible map area are likely to be very unreliable.

Parameters:

coord: {*nokia.maps.geo.Coordinate*}

A point object containing WGS84 coordinates to be translated into a pixel position

Returns:

{*nokia.maps.util.Point*}

An object containing the x and y pixel coordinates relative to the top left corner of the current viewport

getBestZoomLevel(bBoxes): {Number}

This method computes the minimum zoom level at which each of the given bounding boxes fit into the display area.

Parameters:

bBoxes: {*nokia.maps.geo.BoundingBox*[]}

A list of bounding boxes.

Returns:

{Number} the computed zoom level

here

getBoundingBox(): {[*nokia.maps.geo.BoundingBox*](#)}

This method calculates the outer bounding box of all map objects in the given `Display` instance, including all child containers. If the display does not contain any objects or child containers, the method returns `null`. If the display contains only one object that has no geographical dimensions such as a [*nokia.maps.map.Marker*](#) or [*nokia.maps.map.StandardMarker*](#), then the returned bounding box may have the size of zero.

Note that the method does not return the bounding box of the current viewport. For this purpose, please use the method [*nokia.maps.map.Display#getViewBounds*](#).

Returns:

{[*nokia.maps.geo.BoundingBox*](#)}

The calculated geographic outer bounding box of all map objects in the display or `null`, if the display does not contain any map objects

See: [*nokia.maps.map.Display#getViewBounds*](#)

getCam(): {[*nokia.maps.map.ICam*](#)}

This method returns the map camera

Returns:

{[*nokia.maps.map.ICam*](#)}

getComponentById(id): {[*nokia.maps.map.component.Component*](#)}

This method returns the first component with the given identifier or `null` if no component with such an identifier is currently attached to the `Display` instance.

Parameters:

id: {`String`}

The identifier of the component to retrieve

Returns:

{[*nokia.maps.map.component.Component*](#)}

here

The first component from the list of components with the given identifier or null if no component with such an identifier is currently attached to the Display instance

getObjectAt(x, y): {*nokia.maps.map.Object* | undefined}

A method to obtain the topmost visible object at the given position relative to the viewport's origin.

Parameters:

x:	{Number}
	The X coordinate of the pixel position
y:	{Number}
	The Y coordinate of the pixel position

Returns:

{*nokia.maps.map.Object*
| undefined}

the top most object found at the given position or undefined if no object has been found

getObjectsAt(x, y): {*nokia.maps.map.Object*[]}

A method to obtain all objects that are visible at the given pixel position relative to the viewport's origin. The list is sorted according to the drawing order of the objects. The top most object has an index of 0 in returned array.

Parameters:

x:	{Number}
	The X-coordinate of the pixel position
y:	{Number}
	The Y-coordinate of the pixel position

Returns:

here

`{nokia.maps.map.Object[]}`

the list of map objects found at the given pixel position

getObjectsWithin(left, top, right, bottom): `{nokia.maps.map.Object[]}`

This method returns all objects found within a rectangle defined in terms of the pixel coordinates of its top left and bottom right corners.

Parameters:

left: {Number}

The left edge of the rectangle as a number of pixels relative to viewport origin

top: {Number}

The top edge of the rectangle as a number of pixels relative to viewport origin

right: {Number}

The right edge of the rectangle as a number of pixels relative to viewport origin

bottom: {Number}

The bottom edge of the rectangle as a number of pixels relative to viewport origin

Returns:

`{nokia.maps.map.Object[]}`

A list of map objects found within the given rectangle

getView(): `{nokia.maps.map.IView}`

This method returns the map view

Returns:

`{nokia.maps.map.IView}`

here

getViewBounds(): {*nokia.maps.geo.BoundingBox*}

This method retrieves the outer bounding box of the map view (the smallest bounding box covering all visible points).

Returns:

{*nokia.maps.geo.BoundingBox*}

The outer bounding box of the map view.

getZoom(x, y): {Number}

This method returns zoom level at provided screen position.

Parameters:

x: {Number} [optional]

The x-component of the screen coordinate.

y: {Number} [optional]

The y-component of the screen coordinate.

Returns:

{Number} a zoom level at the given screen position

static **interact**(timeStamp, x, y, biX, biY)

This method processes an interaction based on the screen coordinates of the viewport. If `beginInteraction()` was triggered with a `modifiers` argument of `COORD` combined with any other modifier, but `interact` is called without bifocal coordinates (`biX`, `biY`) then the `COORD` modifier is ignored. If no interaction is currently ongoing (`beginInteraction()` has not been called or `endInteraction()` has been called) the call to `interact()` is ignored.

Parameters:

timeStamp: {Number}

The time stamp of interaction needed for computation of the kinetic effect.

here

x:	{Number}	The X coordinate of the interaction.
y:	{Number}	The Y coordinate of the interaction.
biX:	{Number} [optional]	The second X coordinate of the interaction, only needed for bifocal interaction.
biY:	{Number} [optional]	The second Y coordinate of the interaction, only needed for bifocal interaction.

pan(startX, startY, endX, endY, animation)

This method pans the map from the start point to the end point specified by the caller. Both points are defined in terms of screen coordinates. The effect is that the location under the end point moves to the screen coordinates given by the start point. This operation may use a platform specific animation if indicated by the corresponding optional animation string.

Parameters:

startX:	{Number}	The x-position of the pixel relative to the top-left corner of the current view, indicating the x-coordinate of the point from which to start panning
startY:	{Number}	The y-position of the pixel relative to the top-left corner of the current view, indicating the y-coordinate of the point from which to start panning
endX:	{Number}	The x-position of the pixel relative to the top-left corner of the current view, indicating the x-coordinate of the point to which to pan
endY:	{Number}	

here

The y-position of the pixel relative to the top-left corner of the current view, indicating the y-coordinates of the point to which to pan

animation: {String} [optional]

The animation to be used while modifying the view, must be a value from the list of animation types [nokia.maps.map.Display#animation](#)

pixelToGeo(x, y): {[nokia.maps.geo.Coordinate](#)}

This method translates a pixel position within the viewport to geo coordinates. The pixel position is relative to the top left corner of the viewport, (0,0), which is also the top left corner of the visible map. Note: Result values outside the visible map area are likely to be very unreliable.

Parameters:

x: {Number}

The x-coordinate of the pixel position to be translated into longitude.

y: {Number}

The y-coordinate of the pixel position to be translated into latitude.

Returns:

{[nokia.maps.geo.Coordinate](#)}

A point object containing the WGS84 coordinates of the caller-supplied pixel position.

removeComponent(component): {Number}

This method decrements the reference counter of the supplied component and if the counter has reached zero, the method removes the component from display.

Parameters:

component: {[nokia.maps.map.component.Component](#)}

A reference to the component to be removed.

Returns:

`{Number}` The value of the reference counter of the component after the remove call (the element is not removed if this value is greater than zero)

Example:

```
display.removeComponent(display.getComponentById("zoom.DoubleClick"));
```

setAttributes(animation, center, level, tilt, heading)

This method sets a number of properties of the map. The properties include animation, map center, zoom level, map tilt, and map bearing. You must provide actual values only for those properties you wish to change, and undefined for those that should not be altered.

Parameters:

animation: `{String}`

The animation to be used while modifying the map attributes; must be a value from the list of animation types [nokia.maps.map.Display#animation](#)

center: `{nokia.maps.geo.Coordinate}`

The new map center given as an object containing geographical coordinates

level: `{Number}`

The new zoom level given as a value between [nokia.maps.map.Display#minZoomLevel](#) and [nokia.maps.map.Display#maxZoomLevel](#)

tilt: `{Number}`

The new value of map tilt in degrees given as a value between [nokia.maps.map.Display#minTilt](#) and [nokia.maps.map.Display#maxTilt](#)

heading: `{Number}`

The new map heading (bearing) given as a value between [nokia.maps.map.Display#minHeading](#) and [nokia.maps.map.Display#maxHeading](#)

here

setBaseMapType(provider, animate, view)

This method sets base map type provider and optional transition between old and new provider can be involved. There should be also defined view or camera for new base map type so together with animation there should be smooth transition between old base map type and old view/camera and new ones.

Parameters:

provider: {[nokia.maps.map.provider.Provider](#)}

One of the available providers at
nokia.maps.map.Display.availableBaseMapTypes

animate: {String} [optional]

the animation to be used while modifying the view; must be a value from the
list of animation types [nokia.maps.map.Display#animation](#)

view: {[nokia.maps.map.IView](#)} [optional]

New view for new base map type

setCam(cam, animation): {[nokia.maps.map.ICam](#)}

This method sets the new camera for the map.

Parameters:

cam: {[nokia.maps.map.ICam](#)}

New camera

animation: {String} [optional]

The animation to be used while modifying the camera; must be a value from
the list of animation types [nokia.maps.map.Display#animation](#)

Returns:

{[nokia.maps.map.ICam](#)}

Returns adjusted camera to closest available camera.

here

setCenter(coord, animation)

This method centers the map on the location specified by the caller. The method may use a platform specific animation if indicated by the corresponding optional animation string.

Parameters:

coord:	<code>{nokia.maps.geo.Coordinate}</code>
	The location on which to center the map specified in terms of WGS84 coordinates
animation:	<code>{String}</code> [optional]
	The animation to be used while modifying the map center; must be a value from the list of animation types nokia.maps.map.Display#animation

setCopyrightAlignment(alignment)

This method sets the position of the copyright information.

Parameters:

alignment:	<code>{String}</code>
	The alignment relative to the map viewport, where the copyright is to be placed.

See: [nokia.maps.map.Display#copyrightAlignment](#)

setHeading(heading, animation)

This method sets the heading (bearing) of the map. The method may use a platform-specific animation if this is indicated by the corresponding optional animation string.

Parameters:

heading:	<code>{Number}</code>

here

The new value of map heading (bearing) as a number of degrees; a value between [nokia.maps.map.Display#minHeading](#) and [nokia.maps.map.Display#maxHeading](#)

animation: {String} [optional]

The animation to be used while modifying the heading; must be a value from the list of animation types [nokia.maps.map.Display#animation](#).

setPadding(padding1, padding2, padding3, padding4)

This method sets the property [nokia.maps.map.Display#padding](#). The method accepts up to four arguments which provide padding values in the following order: top, right, bottom, left.

The caller must supply at least one argument and optionally up to four as shown in these examples:

- `setPadding(topRightBottomLeft);` - one argument/value, indicates identical padding all round
- `setPadding(topAndBottom, rightAndLeft);` - the first argument sets top and bottom padding, the second right and left padding
- `setPadding(top, rightAndLeft, bottom);` - the second argument sets right and left padding, the first and third arguments set top and bottom padding, respectively
- `setPadding(top, right, bottom, left);` - the first argument sets top padding, the second right padding, the third bottom padding, and the fourth left padding

Parameters:

padding1: {Number}

Top padding in pixels; if this is the only argument provided, it is used also to set right, bottom and left padding (padding is identical all round); if the caller provides only two arguments, padding1 sets both top and bottom padding

padding2: {Number} [optional]

Right padding in pixels; if padding4 (left padding) is not specified by the caller, this argument determines both right and left padding

padding3: {Number} [optional]

Bottom padding in pixels; if not provided, its value is the same as that of padding1 (top padding)

here

padding4: {Number} [optional]

Left padding in pixels; if not provided, its value is supplied by padding2 (right padding) or padding1 (top padding) if padding2 is not provided

setTilt(tilt, animation)

This method sets the tilt of the map. It may use a platform-specific animation if indicated by the corresponding optional animation string.

Parameters:

tilt: {Number}

The new value of map tilt in degrees, a value between [*nokia.maps.map.Display#minTilt*](#) and [*nokia.maps.map.Display#maxTilt*](#).

animation: {String} [optional]

The animation to be used while modifying the tilt; must be a value from the list of animation types [*nokia.maps.map.Display#animation*](#).

setView(view, animation): {[*nokia.maps.map.IView*](#)}

This method sets the new view for the map.

Parameters:

view: {[*nokia.maps.map.IView*](#)}

New view

animation: {String} [optional]

The animation to be used while modifying the view; must be a value from the list of animation types [*nokia.maps.map.Display#animation*](#)

Returns:

{[*nokia.maps.map.IView*](#)}

Returns adjusted view to closest available view.

here

setZoomLevel(level, animation, toX, toY)

This method sets the zoom level to the value specified by the caller. The caller must provide the zoom level and can also supply the animation type and the screen coordinates of a point on which to zoom in (via the arguments toX and toY). This point may correspond, for example, to the mouse position at the time when the user scrolls the mouse wheel to zoom in or out.

Parameters:

level: {Number}

The new zoom level to be set, a value between [nokia.maps.map.Display#minZoomLevel](#) and [nokia.maps.map.Display#maxZoomLevel](#)

animation: {String} [optional]

The animation to be used while modifying the zoom level; must be a value from the list of animation types [nokia.maps.map.Display#animation](#)

toX: {Number} [optional]

The x-position of the pixel relative to the top-left corner of the current view to stay fixed

toY: {Number} [optional]

The y-position of the pixel relative to the top-left corner of the current view to stay fixed

update(delay, quick): {[nokia.maps.map.Display](#)}

This method causes the current map view to be re-rendered.

Parameters:

delay: {Number} [optional]

The maximum acceptable delay before the next frame is rendered as a number of milliseconds; if the value is zero or less, rendering occurs synchronously or at the earliest opportunity; if the argument is omitted or undefined, the system determines when to carry out re-rendering

quick: {Boolean} [optional, default: false]

here

A Boolean to indicate whether full rendering (`false`) or only quick rendering (`true`) is to be performed; quick rendering uses only cached data, while full rendering may request new data and cause new or modified objects to be rendered; the argument is ignored if `delay` is omitted or undefined

Returns:

`{nokia.maps.map.Display}`

A reference to the given instance of `Display` (`this`).

zoomTo(`boundingBox`, `keepCenter`, `animation`)

This method zooms the map to ensure that the bounding box provided by the caller is visible in its entirety in the map viewport. The caller has the option of retaining the current map center, which may cause the zoom to be adjusted so that the bounding box is visible while the map center remains unchanged. The method may use a platform specific animation if this is indicated by the corresponding optional animation string.

Parameters:

`boundingBox`: `{nokia.maps.geo.BoundingBox}`

The bounding box that is to be visible in its entirety in the map viewport

`keepCenter`: `{Boolean}`

A Boolean indicating whether to keep the map center unchanged (`true`) or if the view center may change (`false`);

`animation`: `{String}` [optional]

The animation to be used; must be a value from the list of animation types [nokia.maps.map.Display#animation](#).

Event Details

basemapchangeend

This event is fired to mark the end of a change from one base map type to another (for example `NORMAL` to `SATELLITE`). Until this event has been detected, the map should ignore calls to methods that alter the map view. See also [nokia.maps.map.Display.TransitionEvent](#).

here

Event Handler Parameters:

event `{nokia.maps.map.Display.TransitionEvent}`

basemapchangestart

This event is fired to mark the start of a change from one base map type to another (for example NORMAL to SATELLITE). After this event has been fired and until `nokia.maps.map.Display#basemapchangeend` has been detected, the map should ignore calls to methods that alter the map view. See also [nokia.maps.map.Display.TransitionEvent](#).

Event Handler Parameters:

event `{nokia.maps.map.Display.TransitionEvent}`

displayready

This event is fired when the map has been initialized and is ready to support interactive events, transitions and animations. No code related to map view manipulation, such as calls to [nokia.maps.map.Display#setZoomLevel](#), [nokia.maps.map.Display#setCenter](#), [nokia.maps.map.Display#setBaseMapType](#), etc., must be allowed to run (although components or markers can be added to the map and other operations that do not affect the map view can be performed). See also [nokia.maps.map.Display.TransitionEvent](#).

Event Handler Parameters:

event `{nokia.maps.map.Display.TransitionEvent}`

mapvalueexceeded

This event is fired if a received value exceeds the permitted range. (see [nokia.maps.map.Display.MapValueExceededEvent](#)).

Event Handler Parameters:

event `{nokia.maps.map.Display.MapValueExceededEvent}`

here

mapviewchange

This event is fired each time after the map has been rendered in response to a mapviewchange event (see [nokia.maps.map.Display.MapViewChangeEvent](#)).

The data property of the event object provides information on all the changes that have been effected. (see [nokia.maps.map.Display.MapViewChangeEvent#data](#)).

Note that a large number of mapviewchange events can be fired within the space of a second, therefore it is advisable to avoid time-consuming operations within mapviewchange listeners.

Event Handler Parameters:

event {[nokia.maps.map.Display.MapViewChangeEvent](#)}

mapviewchangeend

This event is fired when the map display has been rendered and all map view changes have been processed (see also [nokia.maps.map.Display.MapViewChangeEvent](#)). It indicates that you can safely execute more expensive operations to respond to a map view change (for example UI updates).

Event Handler Parameters:

event {[nokia.maps.map.Display.MapViewChangeEvent](#)}

mapviewchangestart

This event is fired immediately after the map view has been changed (see [nokia.maps.map.Display.MapViewChangeEvent](#)). It is fired only for the first change of the map view after the rendering of the map display has been completed.

Event Handler Parameters:

event {[nokia.maps.map.Display.MapViewChangeEvent](#)}

resize

here

This event is fired each time a change of the size of the display is detected. It is fired multiple times during a continuous resize operation (for example, if the display is being resized due to user input).

Note that resizing a map also triggers `mapviewchange` events.

Event Handler Parameters:

event `{nokia.maps.dom.Event}`

resizeend

This event is fired when a resizing operation has finished (when no more changes to the size of the display have been detected for a certain amount of time).

Note that resizing a map also triggers `mapviewchange` events.

Event Handler Parameters:

event `{nokia.maps.dom.Event}`

resizestart

This event is fired immediately after the process of resizing the display has begun and then periodically until the process has completed, when `resizeend` is fired.

Note that resizing a map also triggers `mapviewchange` events.

Event Handler Parameters:

event `{nokia.maps.dom.Event}`

transitionend

This event is fired immediately after a map animation has finished or a transition from one base map type to another (in the latter case the event precedes `nokia.maps.map.Display#basemapchangeend`). Until this event has been detected, the map should ignore calls to methods that alter the map view. See also [nokia.maps.map.Display.TransitionEvent](#).

here

Event Handler Parameters:

event {[nokia.maps.map.Display.TransitionEvent](#)}

transitionstart

This event is fired immediately after a map animation starts or a transition from one base map type to another (in the latter case the event follows [nokia.maps.map.Display#basemapchangestart](#)). After this event was fired and until [nokia.maps.map.Display#transitionend](#) has been detected, the map should ignore calls to methods that alter the map view. See also [nokia.maps.map.Display.TransitionEvent](#).

Event Handler Parameters:

event {[nokia.maps.map.Display.TransitionEvent](#)}

Class: BaseMapChangeEvent

This class is a member of [nokia.maps.map.Display](#).

Class Summary

This class represents an event used to signal state of switching between base map types.

[For full details, see [nokia.maps.map.Display.BaseMapChangeEvent](#)]

Table 164: Property Summary

Properties
baseMapData: {Object} This property represents an object which contains additional data about event: - old - old base map type - current - newly set base map type

Directly Inherited Properties

Inherited from class [nokia.maps.map.Object.Event](#) :

[display](#), [displayObjects](#), [displayX](#), [displayY](#)

Inherited from class [nokia.maps.dom.Event](#) :

here

AT_TARGET, bubbles, BUBBLING_PHASE, canBubble, cancelable, canSicker, CAPTURING_PHASE, currentTarget, defaultPrevented, eventPhase, namespaceURI, nativeEvent, page, propagation, PROPAGATION_OK, PROPAGATION_STOP, PROPAGATION_STOP_IMMEDIATE, target, timeStamp, type

Method Summary

Directly Inherited Methods

Inherited from class *nokia.maps.dom.Event* :

cancel, clone, preventDefault, preventUnload, stopImmediatePropagation, stopPropagation

Class Description

This class represents an event used to signal state of switching between base map types.

Constructor Details

nokia.maps.map.Display.BaseMapChangeEvent(defaults)

This method initializes a new instance of BaseMapChangeEvent.

Parameters:

defaults: {Object} [optional]
 see constructor parameter of *nokia.maps.dom.Event*

Property Details

baseMapData: {Object}

This property represents an object which contains additional data about event:

- old - old base map type
- current - newly set base map type

Class: DisplayReadyEvent

This class is a member of *nokia.maps.map.Display*.

Class Summary

This class represents an event used to signal the moment in which map is fully functional and ready to use after initialization

here

[For full details, see [nokia.maps.map.Display.DisplayReadyEvent](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.Object.Event](#) :

[display](#), [displayObjects](#), [displayX](#), [displayY](#)

Inherited from class [nokia.maps.dom.Event](#) :

[AT_TARGET](#), [bubbles](#), [BUBBLING_PHASE](#), [canBubble](#), [cancelable](#), [canSicker](#), [CAPTURING_PHASE](#), [currentTarget](#), [defaultPrevented](#), [eventPhase](#), [namespaceURI](#), [nativeEvent](#), [page](#), [propagation](#), [PROPAGATION_OK](#), [PROPAGATION_STOP](#), [PROPAGATION_STOP_IMMEDIATE](#), [target](#), [timeStamp](#), [type](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.dom.Event](#) :

[cancel](#), [clone](#), [preventDefault](#), [preventUnload](#), [stopImmediatePropagation](#), [stopPropagation](#)

Class Description

This class represents an event used to signal the moment in which map is fully functional and ready to use after initialization

Constructor Details

`nokia.maps.map.Display.DisplayReadyEvent(defaults)`

This method initializes a new instance of `DisplayReadyEvent`.

Parameters:

`defaults`: {Object} [optional]

see constructor parameter of [nokia.maps.dom.Event](#)

Class: *MapValueExceededEvent*

This class is a member of [nokia.maps.map.Display](#).

here

Class Summary

This class represents an event used to signal that value assigned to the property is out of supported range Setting the following values can trigger this event:

- [nokia.maps.map.Display#zoomLevel](#)
- [nokia.maps.map.IView#zoom](#)
- [nokia.maps.map.ICam#fov](#)

[For full details, see [nokia.maps.map.Display.MapValueExceededEvent](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.Object.Event](#) :

[display](#), [displayObjects](#), [displayX](#), [displayY](#)

Inherited from class [nokia.maps.dom.Event](#) :

[AT_TARGET](#), [bubbles](#), [BUBBLING_PHASE](#), [canBubble](#), [cancelable](#), [canSicker](#), [CAPTURING_PHASE](#), [currentTarget](#), [defaultPrevented](#), [eventPhase](#), [namespaceURI](#), [nativeEvent](#), [page](#), [propagation](#), [PROPAGATION_OK](#), [PROPAGATION_STOP](#), [PROPAGATION_STOP_IMMEDIATE](#), [target](#), [timeStamp](#), [type](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.dom.Event](#) :

[cancel](#), [clone](#), [preventDefault](#), [preventUnload](#), [stopImmediatePropagation](#), [stopPropagation](#)

Class Description

This class represents an event used to signal that value assigned to the property is out of supported range Setting the following values can trigger this event:

- [nokia.maps.map.Display#zoomLevel](#)
- [nokia.maps.map.IView#zoom](#)
- [nokia.maps.map.ICam#fov](#)

Constructor Details

`nokia.maps.map.Display.MapValueExceededEvent`(defaults)

here

This method initializes a new instance of `MapValueExceededEvent`.

Parameters:

`defaults`: `{Object}` [optional]
 see constructor parameter of [nokia.maps.dom.Event](#)

Class: `MapViewChangeEvent`

This class is a member of [nokia.maps.map.Display](#).

Class Summary

This class represents an event that signals changes to the map view.

[For full details, see [nokia.maps.map.Display.MapViewChangeEvent](#)]

Table 165: Property Summary

Properties
<code>data</code>: <code>{Number}</code> This bitmask property contains cumulative information about the map view properties which were altered during display rendering.
static <code>MAPVIEWCHANGE_CENTER</code>: <code>{Number}</code> This constant represents a bit flag that indicates a change in coordinates of the map center.
static <code>MAPVIEWCHANGE_HEADING</code>: <code>{Number}</code> This constant represents a bit flag that indicates a change in the heading (bearing) of the map.
static <code>MAPVIEWCHANGE_INCLINE</code>: <code>{Number}</code> This constant represents a bit flag that indicates a change of the incline.
static <code>MAPVIEWCHANGE_SIZE</code>: <code>{Number}</code> This constant represents a bit flag that indicates a change in the size of the display.
static <code>MAPVIEWCHANGE_TILT</code>: <code>{Number}</code> This constant represents a bit flag that indicates a change in the tilt of the map.
static <code>MAPVIEWCHANGE_ZOOM</code>: <code>{Number}</code> This constant represents a bit flag that indicates a change in the map zoom level.
<code>valueData</code>: <code>{Object}</code> This property represents an object which contains additional data about event: • <code>propertyName</code> - name of the property that have been modified, • <code>value</code> - value passed to the property,

here

Properties

- `min` - minimal allowed value for the property,
- `max` - maximum value of the property

Directly Inherited Properties

Inherited from class [*nokia.maps.map.Object.Event*](#) :

[*display*](#), [*displayObjects*](#), [*displayX*](#), [*displayY*](#)

Inherited from class [*nokia.maps.dom.Event*](#) :

[*AT_TARGET*](#), [*bubbles*](#), [*BUBBLING_PHASE*](#), [*canBubble*](#), [*cancelable*](#), [*canSicker*](#), [*CAPTURING_PHASE*](#), [*currentTarget*](#), [*defaultPrevented*](#), [*eventPhase*](#), [*namespaceURI*](#), [*nativeEvent*](#), [*page*](#), [*propagation*](#), [*PROPAGATION_OK*](#), [*PROPAGATION_STOP*](#), [*PROPAGATION_STOP_IMMEDIATE*](#), [*target*](#), [*timeStamp*](#), [*type*](#)

Method Summary

Directly Inherited Methods

Inherited from class [*nokia.maps.dom.Event*](#) :

[*cancel*](#), [*clone*](#), [*preventDefault*](#), [*preventUnload*](#), [*stopImmediatePropagation*](#), [*stopPropagation*](#)

Class Description

This class represents an event used to signal changes of the map view. A map view is defined by the following display properties:

- [*nokia.maps.map.Display.Properties#center*](#)
- [*nokia.maps.map.Display.Properties#zoomLevel*](#)
- [*nokia.maps.map.Display.Properties#heading*](#)
- [*nokia.maps.map.Display.Properties#tilt*](#)
- [*nokia.maps.map.Display#width*](#)
- [*nokia.maps.map.Display#height*](#)

Constructor Details

`nokia.maps.map.Display.MapViewChangeEvent`(defaults)

This method initializes a new instance of `MapViewChangeEvent`.

Parameters:

here

defaults: {Object} [optional]

see constructor parameter of [nokia.maps.dom.Event](#)

Property Details

data: {Number}

This bitmask property contains cumulative information about the map view properties which were altered during display rendering. It is only set for `nokia.maps.map.Display#mapviewchange`, otherwise it is always 0.

Example:

```
// Use bitwise "&" operator, to check for specific map view property
// changes:
map.addListener("mapviewchange", function (event) {
    if (event.data & event.MAPVIEWCHANGE_CENTER) {
        console.log("map view changed: center");
    }
});
```

static **MAPVIEWCHANGE_CENTER:** {Number}

This constant represents a bit flag that indicates a change in coordinates of the map center.

See: [nokia.maps.map.Display.MapViewChangeEvent#data](#)

static **MAPVIEWCHANGE_HEADING:** {Number}

This constant represents a bit flag that indicates a change in the heading (bearing) of the map.

See: [nokia.maps.map.Display.MapViewChangeEvent#data](#)

static **MAPVIEWCHANGE_INCLINE:** {Number}

This constant represents a bit flag that indicates a change of the incline.

See: [nokia.maps.map.Display.MapViewChangeEvent#data](#)

static **MAPVIEWCHANGE_SIZE:** {Number}

here

This constant represents a bit flag that indicates a change in the size of the display. The size of the display is defined by the properties [nokia.maps.map.Display#width](#) and [nokia.maps.map.Display#height](#)

See: [nokia.maps.map.Display.MapViewChangeEvent#data](#)

static **MAPVIEWCHANGE_TILT**: {Number}

This constant represents a bit flag that indicates a change in the tilt of the map.

See: [nokia.maps.map.Display.MapViewChangeEvent#data](#)

static **MAPVIEWCHANGE_ZOOM**: {Number}

This constant represents a bit flag that indicates a change in the map zoom level.

See: [nokia.maps.map.Display.MapViewChangeEvent#data](#)

valueData: {Object}

This property represents an object which contains additional data about event:

- **propertyName** - name of the property that have been modified,
- **value** - value passed to the property,
- **min** - minimal allowed value for the property,
- **max** - maximum value of the property

Interface: Properties

This interface is a member of [nokia.maps.map.Display](#).

Interface Summary

This interface defines the properties (keys) that can be passed to the map [nokia.maps.map.Display](#) constructor.

[For full details, see [nokia.maps.map.Display.Properties](#)]

here

Table 166: Property Summary

Properties
baseMapType: {nokia.maps.map.provider.Provider} This property holds the initial base map type which is used to display the map.
center: {nokia.maps.geo.Coordinate} This property contains the initial coordinates of the map center.
components: {nokia.maps.map.component.Component[][]} The array of all components that should be added to the map (to the given instance of nokia.maps.map.Display).
fading: {Number} This property contains the initial value indicating the length of time (in milliseconds) over which new asynchronously provided tiles are faded into the display.
fixedCenter: {Boolean} This property contains the initial value for flag indicating if the center of the map should remain unchanged if the display is resized or if padding changes.
heading: {Number} This property holds initial heading of the current view.
margin: {Number} This property contains the initial size of the supplemental rendering area.
tilt: {Number} This property holds initial tilt of the current view.
view: {nokia.maps.map.IView} This property holds initial view.
viewBounds: {nokia.maps.geo.BoundingBox} This property contains the initial boundingBox to which map should zoom to.
zoomLevel: {Number} This property holds initial zoom level of the current view.

Directly Inherited Properties

Inherited from class [nokia.maps.map.Container.Properties](#) :

[objects](#)

Inherited from class [nokia.maps.map.Object.Properties](#) :

[visibility](#), [zIndex](#)

here

Interface Description

This interface defines the properties (keys) that can be passed to the map [nokia.maps.map.Display](#) constructor.

Property Details

baseMapType: {[nokia.maps.map.provider.Provider](#)}

This property holds the initial base map type which is used to display the map. The value must be one of the display base map type constants.

center: {[nokia.maps.geo.Coordinate](#)}

This property contains the initial coordinates of the map center. There should be used one of these properties: center, view or initialBoundingBox

components: {[nokia.maps.map.component.Component](#)[]} }

The array of all components that should be added to the map (to the given instance of [nokia.maps.map.Display](#)).

fading: {Number}

This property contains the initial value indicating the length of time (in milliseconds) over which new asynchronously provided tiles are faded into the display. The value must be zero or a positive integer, with zero indicating no fade-in.

fixedCenter: {Boolean}

This property contains the initial value for flag indicating if the center of the map should remain unchanged if the display is resized or if padding changes.

heading: {Number}

This property holds initial heading of the current view.

margin: {Number}

here

This property contains the initial size of the supplemental rendering area. The renderer must be prepared to allow rapid mouse panning within the displayed area extended on each side by the given margin (in pixels).

tilt: {Number}

This property holds initial tilt of the current view.

view: {[nokia.maps.map.IView](#)}

This property holds initial view.

viewBounds: {[nokia.maps.geo.BoundingBox](#)}

This property contains the initial boundingBox to which map should zoom to. If this property is set, than center and zoomLevel properties are overwritten by it.

zoomLevel: {Number}

This property holds initial zoom level of the current view.

Class: TransitionEvent

This class is a member of [nokia.maps.map.Display](#).

Class Summary

This class represents an event used to signal state of animation.

[For full details, see [nokia.maps.map.Display.TransitionEvent](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.Object.Event](#) :

[display](#), [displayObjects](#), [displayX](#), [displayY](#)

Inherited from class [nokia.maps.dom.Event](#) :

[AT_TARGET](#), [bubbles](#), [BUBBLING_PHASE](#), [canBubble](#), [cancelable](#), [canSicker](#), [CAPTURING_PHASE](#), [currentTarget](#), [defaultPrevented](#), [eventPhase](#), [namespaceURI](#), [nativeEvent](#), [page](#), [propagation](#),

here

[PROPAGATION_OK](#), [PROPAGATION_STOP](#), [PROPAGATION_STOP_IMMEDIATE](#), [target](#), [timeStamp](#), [type](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.dom.Event](#) :

[cancel](#), [clone](#), [preventDefault](#), [preventUnload](#), [stopImmediatePropagation](#), [stopPropagation](#)

Class Description

This class represents an event used to signal state of animation.

Constructor Details

nokia.maps.map.Display.TransitionEvent(defaults)

This method initializes a new instance of TransitionEvent.

Parameters:

defaults: {Object} [optional]

see constructor parameter of [nokia.maps.dom.Event](#)

Interface: ICam

This interface is a member of [nokia.maps.map](#).

Interface Summary

This interface describes a map camera view.

[For full details, see [nokia.maps.map.ICam](#)]

Table 167: Property Summary

Properties
fov: {Number} The field of view (also field of vision, abbreviated FOV) of the camera in degrees in degrees ranging from 0° to +360°.
pitch: {Number undefined} The vertical orientation of the camera (equivalent to rotation around its right axis) in degrees ranging from 0° (straight downwards) to +180° (straight upwards).

here

Properties

roll: {Number | undefined}

The rotation of the camera around its forward direction in degrees, 0° is upright, positive angle means an inclination to the right, negative to the left.

yaw: {Number | undefined}

The horizontal orientation of the camera in degrees, where 0° is northwards, 90° eastwards, 180° southwards and 270° westwards.

Directly Inherited Properties

Inherited from class [nokia.maps.geo.ICoordinate](#) :

[altitude](#), [altMode](#), [latitude](#), [longitude](#)

Interface Description

This interface describes a map camera view.

Property Details

fov: {Number}

The field of view (also field of vision, abbreviated FOV) of the camera in degrees in degrees ranging from 0° to +360°.

pitch: {Number | undefined}

The vertical orientation of the camera (equivalent to rotation around its right axis) in degrees ranging from 0° (straight downwards) to +180° (straight upwards). It's also known as tilt. A value of undefined is treated as 0.

roll: {Number | undefined}

The rotation of the camera around its forward direction in degrees, 0° is upright, positive angle means an inclination to the right, negative to the left. It's also known as "Dutch tilt" A value of undefined is treated as 0.

yaw: {Number | undefined}

The horizontal orientation of the camera in degrees, where 0° is northwards, 90° eastwards, 180° southwards and 270° westwards. It's also known as heading. A value of undefined is treated as 0.

here

Interface: IHitArea

This interface is a member of [nokia.maps.map](#).

Interface Summary

This interface defines the properties of a hit area.

[For full details, see [nokia.maps.map.IHitArea](#)]

Table 168: Property Summary

Properties
readonly type : {String} The type of the area.
readonly values : {Number[]} The type-dependent values to define the shape of the hit area.

Interface Description

This interface defines the properties of a hit area.

Property Details

readonly **type**: {String}

The type of the area. It has to be one of the strings "rect", "circle" or "poly". See also <http://www.w3.org/TR/REC-html40/struct/objects.html#edef-shape>.

readonly **values**: {Number[]}

The type-dependent values to define the shape of the hit area. The format for the different types are:

- "rect": [left, top, right, bottom]
- "circle": [centerX, centerY, radius]
- "poly": [x1, y1, x2, y2 ... xN,yN]

See also <http://www.w3.org/TR/REC-html40/struct/objects.html#edef-coords>

here

Interface: IView

This interface is a member of [nokia.maps.map](#).

Interface Summary

This interface describes a map view.

[For full details, see [nokia.maps.map.IView](#)]

Table 169: Property Summary

Properties
distance: {Number} The distance from where the view point is seen in meters.
heading: {Number} The horizontal orientation from where the view point is seen, 0° means from south, 90° from west, 180° from north and 270° from east.
incline: {Number} The lateral incline of the view point, 0° is upright, positive angle means an incline to the left, negative to the right.
tilt: {Number} The vertical orientation from where the view point is seen in degrees, ranging from 0° (seen from above) to +180° (seen from below).
zoom: {Number} The zoom level of the view.

Directly Inherited Properties

Inherited from class [nokia.maps.geo.ICoordinate](#) :

[altitude](#), [altMode](#), [latitude](#), [longitude](#)

Interface Description

This interface describes a map view.

Property Details

[distance](#): {Number}

The distance from where the view point is seen in meters. A value NaN indicates that zoom level should be used to compute distance.

here

Default Value: NaN

heading: {Number}

The horizontal orientation from where the view point is seen, 0° means from south, 90° from west, 180° from north and 270° from east. A value of undefined is treated as 0.

incline: {Number}

The lateral incline of the view point, 0° is upright, positive angle means an incline to the left, negative to the right. A value of undefined is treated as 0.

tilt: {Number}

The vertical orientation from where the view point is seen in degrees, ranging from 0° (seen from above) to +180° (seen from below). A value of undefined is treated as 0.

zoom: {Number}

The zoom level of the view. A zoom level defines the size of the (theoretical rolled out) rendered equator viewed vertical. The formula to translate a zoom level into an equator size is: `size = Math.pow(2, zoom) * 256`; The formula to translate an equator size into a zoom level is: `zoom = Math.log( size / 256 ) / Math.LN2`;

Class: Marker

This class is a member of [nokia.maps.map](#).

Extends: [nokia.maps.map.Object](#)

Class Summary

This class represents marker, which representing one geographical coordinate on the map.

[For full details, see [nokia.maps.map.Marker](#)]

Table 170: Property Summary

Properties
anchor: { nokia.maps.util.IPoint }

here

Properties

The anchor point of this marker.

coordinate: {*nokia.maps.geo.Coordinate*}

The coordinates the marker points to.

hitArea: {*nokia.maps.map.IHitArea*}

The (optional) hit area of the Marker.

icon: {*nokia.maps.gfx.Image* | Image | String}

The image that indicates the marker's location on the map (makes the marker visible).

Directly Inherited Properties

Inherited from class *nokia.maps.map.Object* :

CHANGE_SPATIAL, *CHANGE_VISUAL*, *CHANGE_ZINDEX*, *visibility*, *zIndex*

Inherited from class *nokia.maps.dom.EventTarget* :

isDraggable, *isEventTarget*, *parentNode*, *parentNodes*

Inherited from class *nokia.maps.map.provider.IData* :

id

Table 171: Method Summary

Methods

getDisplayBoundingBox (display) : {*nokia.maps.util.Rectangle*}

This method retrieves the bounding box of the marker icon for the display specified by the caller.

getDisplayOffset (display, x, y)

The method calculates the pixel offset between the screen coordinates supplied by the caller and those of the given marker.

getIconForRendering (doc) : {*nokia.maps.gfx.Image*}

The method returns the icon for the given marker bound to the document specified by the caller.

hitTest (display, x, y, tolerance)

The method tests if the given marker (its hit area) covers the location specified by the caller.

Directly Inherited Methods

Inherited from class *nokia.maps.map.Object* :

destroy, *getBoundingBox*, *getDisplay*, *getParent*, *getProvider*, *getRoot*, *isVisible*

Inherited from class *nokia.maps.util.OObject* :

addObserver, *get*, *remove*, *removeObserver*, *set*

here

Inherited from class *nokia.maps.dom.EventTarget* :

addListener, *addListenerNS*, *addListeners*, *disableDrag*, *disableUserSelect*, *dispatch*, *enableDrag*, *enableUserSelect*, *hitTest*, *insertListener*, *insertListenerNS*, *removeAllListeners*, *removeListener*, *removeListenerNS*

Inherited from class *nokia.maps.map.provider.IData* :

getInvalidations

Event Summary

Directly Inherited Events

Inherited from class *nokia.maps.dom.MouseEventTarget* :

click, *dblclick*, *longpress*, *mousedown*, *mouseenter*, *mouseleave*, *mousemove*, *mouseout*, *mouseover*, *mouseup*, *mousewheel*

Inherited from class *nokia.maps.dom.DragEventTarget* :

drag, *dragend*, *dragenter*, *dragleave*, *dragover*, *dragstart*, *drop*

Inherited from class *nokia.maps.dom.TouchEventTarget* :

dbltap, *gesturechange*, *gestureend*, *gesturestart*, *longpress*, *tap*, *touchend*, *touchmove*, *touchstart*

Class Description

This class represents marker, which offers a means of identifying a location on the map with an icon. A marker can be visible or hidden, draggable or fixed. It can also be grouped together with other markers in a container. Markers can be event targets and can be made to react to events. Changes in their state can also be observed by registered observer objects.

Constructor Details

nokia.maps.map.Marker(coord, props)

This method creates a marker object.

Parameters:

table border="0">
coord:	*{nokia.maps.geo.Coordinate}*
	An object containing the geographical coordinates of the marker
props:	*{nokia.maps.map.Marker.Properties}* [optional]

here

An object containing the initial values of marker properties

Property Details

anchor: `{nokia.maps.util.IPoint}`

The anchor point of this marker. The marker's icon is positioned on the map in relation to this anchor point. By default the anchor references the top left corner of the image.

coordinate: `{nokia.maps.geo.Coordinate}`

The coordinates the marker points to.

hitArea: `{nokia.maps.map.IHitArea}`

The (optional) hit area of the Marker.

Default Value: undefined

icon: `{nokia.maps.gfx.Image | Image | String}`

The image that indicates the marker's location on the map (makes the marker visible). NOTE: Either a URL of a bitmap image or an SVG mark-up string can be used when passing a {String} as the value of this property.

Method Details

getDisplayBoundingBox(display): `{nokia.maps.util.Rectangle}`

This method retrieves the bounding box of the marker icon for the display specified by the caller.

Parameters:

display: `{nokia.maps.map.Display}`

The display for which the bounding box of the marker should be determined

Returns:

`{nokia.maps.util.Rectangle}`

here

A rectangle object containing the pixel dimensions on the marker or `null` if the marker is not part of the display specified by the caller or if the marker has no icon

getDisplayOffset(display, x, y)

The method calculates the pixel offset between the screen coordinates supplied by the caller and those of the given marker.

Parameters:

display: {*nokia.maps.map.Display*}

The display for which the offset should be calculated

x: {Object}

The x position as a number of pixels from the top-left corner of the display

y: {Object}

The y position as a number of pixels from the top-left corner of the display

Returns:

An object (*nokia.maps.util.Point*) that represents the difference between the screen coordinates supplied by the caller and the screen coordinates of the given marker, or `null` if the marker is not visible in the specified display

getIconForRendering(doc): {*nokia.maps.gfx.Image*}

The method returns the icon for the given marker bound to the document specified by the caller.

Parameters:

doc: {Document} [optional, default: document]

The document to which the given marker is attached

Returns:

{*nokia.maps.gfx.Image*}

here

The icon image for the marker bound to the given document

hitTest(display, x, y, tolerance)

The method tests if the given marker (its hit area) covers the location specified by the caller. If no hit area (`map.Marker.hitArea`) is defined, the entire screen bounding box within which of the marker appears is taken into account.

Parameters:

display: {*nokia.maps.map.Display*}

The display within which to carry out the check

x: {Number}

The X component of the screen coordinate in pixels

y: {Number}

The Y component of the screen coordinate in pixels

tolerance: {Number} [optional, default: 0]

The tolerance to use for hit testing as a number of pixels

Returns:

A {Boolean} whose value indicates if the given marker covers the specified location (`true`) or not (`false`)

Interface: Properties

This interface is a member of *nokia.maps.map.Marker*.

Interface Summary

This interface defines the properties (keys) that can be passed to the constructor of all classes derived from *nokia.maps.map.Marker*.

[For full details, see *nokia.maps.map.Marker.Properties*]

here

Table 172: Property Summary

Properties
anchor : { nokia.maps.util.IPoint } The initial anchor point of this marker.
draggable : {Boolean} This flag indicates if Marker will be draggable or not.
hitArea : { nokia.maps.map.IHitArea } The initial hit area of the created Marker.
icon : { nokia.maps.gfx.Image Image String} The image that indicates the marker's location on the map (makes the marker visible).

Directly Inherited Properties

Inherited from class [nokia.maps.map.Object.Properties](#) :

[visibility](#), [zIndex](#)

Interface Description

This interface defines the properties (keys) that can be passed to the constructor of all classes derived from [nokia.maps.map.Marker](#).

Property Details

anchor: {[nokia.maps.util.IPoint](#)}

The initial anchor point of this marker. The marker's icon is positioned on the map in relation to this anchor point. By default the anchor references the top left corner of the image.

draggable: {Boolean}

This flag indicates if Marker will be draggable or not.

hitArea: {[nokia.maps.map.IHitArea](#)}

The initial hit area of the created Marker.

icon: {[nokia.maps.gfx.Image](#) | Image | String}

here

The image that indicates the marker's location on the map (makes the marker visible). NOTE: Either a URL of a bitmap image or an SVG mark-up string can be used when passing a {String} as the value of this property. If omitted a standardized icon is used.

Class: Object

This class is a member of *nokia.maps.map*.

Extends: *nokia.maps.dom.EventTarget*, *nokia.maps.map.provider.IData*, *nokia.maps.util.OObject*

Class Summary

Object is the abstract base class for the map itself and for all visual objects that can be displayed in the map.

[For full details, see *nokia.maps.map.Object*]

Table 173: Property Summary

Properties
<i>CHANGE_SPATIAL</i> : {Number} This property represents the invalidation type for spatial changes.
readonly static <i>CHANGE_SPATIAL</i> : {Number} This constant represents the invalidation type for spatial changes.
readonly static <i>CHANGE_VISUAL</i> : {Number} This constant represents the invalidation type for visual changes.
<i>CHANGE_VISUAL</i> : {Number} This property represents the invalidation type for visual changes.
<i>CHANGE_ZINDEX</i> : {Number} This property represents the invalidation type for z-index changes.
readonly static <i>CHANGE_ZINDEX</i> : {Number} This constant represents the invalidation type for z-index changes.
<i>visibility</i> : {Boolean} This property is a flag that determines the visibility of the given Object instance.
<i>zIndex</i> : {Number} This property represents the z-index relative to the container.

Directly Inherited Properties

Inherited from class *nokia.maps.dom.EventTarget* :

here

isDraggable, isEventTarget, parentNode, parentNodes

Inherited from class *nokia.maps.map.provider.IData* :

id

Table 174: Method Summary

Methods
<i>destroy</i> () This method cleans the given instance of <code>Object</code> and removes it from <code>Providers</code> to which it is connected.
<i>getBoundingBox</i> () : {<i>nokia.maps.geo.BoundingBox</i>} This method calculates the outer geographic bounding box of the given object.
<i>getDisplay</i> () : {<i>nokia.maps.map.Display</i>} This method returns the <code>Display</code> instance to which it is attached through the object hierarchy or null if it is not attached to any display.
<i>getParent</i> () : {<i>nokia.maps.map.Object</i>} This method returns an instance of <code>map.Container</code> in which the map object is attached.
<i>getProvider</i> () : {<i>nokia.maps.map.provider.Provider</i>} This method retrieves the instances of <code>map.provider.Provider</code> to which the given object is connected.
<i>getRoot</i> () : {<i>nokia.maps.map.Object</i>} This method returns the root object in which the map object is attached or the object itself.
<i>isVisible</i> () This method checks if the given map object is visible.

Directly Inherited Methods

Inherited from class *nokia.maps.util.OObject* :

addObserver, get, remove, removeObserver, set

Inherited from class *nokia.maps.dom.EventTarget* :

addListener, addListenerNS, addListeners, disableDrag, disableUserSelect, dispatch, enableDrag, enableUserSelect, hitTest, insertListener, insertListenerNS, removeAllListeners, removeListener, removeListenerNS

Inherited from class *nokia.maps.map.provider.IData* :

getInvalidations

here

Event Summary

Directly Inherited Events

Inherited from class *nokia.maps.dom.MouseEventTarget* :

click, dblclick, longpress, mousedown, mouseenter, mouseleave, mousemove, mouseout, mouseover, mouseup, mousewheel

Inherited from class *nokia.maps.dom.DragEventTarget* :

drag, dragend, dragenter, dragleave, dragover, dragstart, drop

Inherited from class *nokia.maps.dom.TouchEventTarget* :

dbltap, gesturechange, gestureend, gesturestart, longpress, tap, touchend, touchmove, touchstart

Class Description

Object is the abstract base class for the map itself and for all visual objects that can be displayed in the map. It is responsible for identifying UI state changes and automatically triggering a redraw. Redraws are platform specific in terms of how and when they are performed.

Object instance can be attached to only one map in given time. When object is added to container the container checks that given Object is 'free to use', e.g. not attached in any map. If so Object is removed from previous map instance (e.g. parent container) and than attached to new one parent container.

Constructor Details

nokia.maps.map.Object(props)

This method creates an instance of Object, which is the base class for the map as well as all visual objects shown in the map.

Parameters:

props: *{nokia.maps.map.Object.Properties}* [optional]

An object providing the initial values of the properties it contains

Property Details

CHANGE_SPATIAL: {Number}

This property represents the invalidation type for spatial changes.

here

`readonly static CHANGE_SPATIAL: {Number}`

This constant represents the invalidation type for spatial changes.

`readonly static CHANGE_VISUAL: {Number}`

This constant represents the invalidation type for visual changes.

`CHANGE_VISUAL: {Number}`

This property represents the invalidation type for visual changes.

`CHANGE_ZINDEX: {Number}`

This property represents the invalidation type for z-index changes.

`readonly static CHANGE_ZINDEX: {Number}`

This constant represents the invalidation type for z-index changes.

`visibility: {Boolean}`

This property is a flag that determines the visibility of the given `Object` instance. An object is visible if this property is `true` and if all its parent objects are visible (see also `nokia.maps.map.Object.isVisible()`).

`zIndex: {Number}`

This property represents the z-index relative to the container. It is used to specify the stacking order within the container.

Objects with the highest values are placed on top (closer to the viewer). The overall stack order is defined hierarchically according to the z-index: objects are ordered within the container to which they belong, and containers are ordered within the `Display` instance. If a container A has a higher z-index than container B, objects that belong to A appear closer to the viewer than any object in B, even if all or some of the objects in B may have a higher z-index than some or all objects in A.

Markers are treated differently than Spatial objects so than `zIndex` property is used for sorting Markers and Spatial object in different layers and marker layer is rendered always above object layer.

here

Method Details

destroy()

This method cleans the given instance of `Object` and removes it from `Providers` to which it is connected. It is always better to call this method when the instance is not needed instead of removing the instance reference only.

getBoundingBox(): `{nokia.maps.geo.BoundingBox}`

This method calculates the outer geographic bounding box of the given object. If the object has no geographic dimensions, `null` is returned. Note that the returned bounding box might have the size of zero, if the object is a one-dimensional object. This happens for example for instances of `nokia.maps.map.Marker` and `nokia.maps.map.StandardMarker`, because they are point objects with a pure visual (pixel base) representation.

Returns:

`{nokia.maps.geo.BoundingBox}`

The calculated geographic outer bounding box object or `null`, if the object has no geographic dimensions

getDisplay(): `{nokia.maps.map.Display}`

This method returns the `Display` instance to which it is attached through the object hierarchy or `null` if it is not attached to any display.

Returns:

`{nokia.maps.map.Display}`

The `Display` instance or `null` if the map object is not attached to any display

getParent(): `{nokia.maps.map.Object}`

This method returns an instance of `map.Container` in which the map object is attached.

Returns:

`{nokia.maps.map.Object}`

here

The parent object or undefined if the map object has no parent (not added to any Provider)

getProvider(): {*nokia.maps.map.provider.Provider*}

This method retrieves the instances of *map.provider.Provider* to which the given object is connected.

Returns:

{*nokia.maps.map.provider.Provider*}

An provider instance to which object is connected

getRoot(): {*nokia.maps.map.Object*}

This method returns the root object in which the map object is attached or the object itself.

Since: 2.5

Returns:

{*nokia.maps.map.Object*}

The root object

isVisible()

This method checks if the given map object is visible. The method returns *true* provided that the object itself is visible, all its parent objects are visible, and if the object is attached to a map (parent chain ends with a map). The returned value is valid for the attached instance of *nokia.maps.map.provider.Provider*.

Returns:

A {Boolean} value, *true* if the given map object is currently visible.

Interface: *Event*

This interface is a member of *nokia.maps.map.Object*.

Extends: *nokia.maps.dom.Event*

here

Interface Summary

This class defines the interface that must be supported by all events fired by a [nokia.maps.map.Object](#) or its derived classes, including the [nokia.maps.map.Display](#) class.

[For full details, see [nokia.maps.map.Object.Event](#)]

Table 175: Property Summary

Properties
display: {nokia.maps.map.Display} This property contains the reference to the display within which the event occurred, if applicable.
displayObjects: {nokia.maps.map.Object[]} This property holds an array of all map objects under the event position reflecting the z-index order of the objects (the first element in the array is the topmost visible object, and the last one is the object with the highest z-index).
displayX: {Number} This property holds the horizontal pixel position where the event occurred relative to the top-left corner of the display associated with the event.
displayY: {Number} This property holds the vertical pixel position where the event occurred relative to the to the top-left corner of the display associated with the event.

Directly Inherited Properties

Inherited from class [nokia.maps.dom.Event](#) :

[AT_TARGET](#), [bubbles](#), [BUBBLING_PHASE](#), [canBubble](#), [cancelable](#), [canSicker](#), [CAPTURING_PHASE](#), [currentTarget](#), [defaultPrevented](#), [eventPhase](#), [namespaceURI](#), [nativeEvent](#), [page](#), [propagation](#), [PROPAGATION_OK](#), [PROPAGATION_STOP](#), [PROPAGATION_STOP_IMMEDIATE](#), [target](#), [timeStamp](#), [type](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.dom.Event](#) :

[cancel](#), [clone](#), [preventDefault](#), [preventUnload](#), [stopImmediatePropagation](#), [stopPropagation](#)

Interface Description

This class defines the interface that must be supported by all events fired by a [nokia.maps.map.Object](#) or its derived classes, including the [nokia.maps.map.Display](#) class. These classes therefore implement this interface.

here

This class does not exist, but is documented here to record the event properties. All events are ultimately instances of the class [nokia.maps.dom.Event](#) and have the properties described here.

Before you continue reading, please ensure that you are familiar with the normal DOM event behavior as explained, for example, in the documentation for the class [nokia.maps.dom.EventTarget](#) or in the corresponding W3C [DOM Event Level 3](#) specification

Dispatch

A [nokia.maps.map.Object.Event](#) has the property `display` to identify the [nokia.maps.map.Display](#) instance in which the event occurred to help establish the correct [propagation path](#). For example, if the user clicks on a marker, a DOM click event with the `map.Display` instance as target is generated by the browser, but because the browser does not know about the markers (at least in some rendering engines), the display itself must find the map objects under the mouse cursor and modify the [propagation path](#). Furthermore, it needs to apply the [nokia.maps.map.Object.Event](#) interface to that DOM event with the special properties `display`, `displayX`, `displayY` and `displayObjects`.

The [nokia.maps.dom.EventTarget#dispatch\(\)](#) method does not generate the [propagation path](#) simply by following the property `parentNode`. Instead, it intercepts the dispatched events, allows the class [nokia.maps.map.Display](#) to find the target and build the [propagation path](#) so that map objects under the cursor are treated like normal DOM nodes.

Separate the data model from the view tree

The modified event dispatch model permits dispatch of visual events that are separate from the data model events. In addition to the hierarchy of map objects within instances of `map.Display`, a further document hierarchy for all map objects may be present and these objects can act as event targets. Therefore, a single data model can be maintained alongside different display-related view models. This is a strength of the event model. While remaining compatible with the DOM event system, it allows you to maintain a document data model alongside the visual representation, using the same objects for the view trees as well as for the data model.

In addition, events can flow over into the normal DOM tree so that you can capture events fired by the display as well as those fired within the normal DOM tree. For example, if a `div` element holds both the map and the UI components, you can use the same listeners to process map events and UI events.

Listener notes

Registering listeners on map objects is similar to registering listeners on DOM nodes, using the class [nokia.maps.dom.EventTarget](#). Note that the listeners are called for all events that reach the map objects. Therefore you must check the property `display` to see if an event is related to a display or a DOM event, and to which `map.Display` instance it belongs.

here

Property Details

display: {[nokia.maps.map.Display](#)}

This property contains the reference to the display within which the event occurred, if applicable.

displayObjects: {[nokia.maps.map.Object](#)[]}

This property holds an array of all map objects under the event position reflecting the z-index order of the objects (the first element in the array is the topmost visible object, and the last one is the object with the highest z-index). Note that the event position can be, for example the mouse cursor position, the user's finger touch position of a touch screen, or any other position related to the event (see [nokia.maps.map.Object.Event#displayX](#) and [nokia.maps.map.Object.Event#displayY](#)).

This property may be undefined, if the event is not related to a display or a specific position, which is the case, for example, for keyboard events, map view change events or some customer defined events.

displayX: {Number}

This property holds the horizontal pixel position where the event occurred relative to the top-left corner of the display associated with the event.

This property may be undefined, if the event is not related to a display or a specific position, which is the case, for example, for keyboard events, map view change events or some customer defined events.

displayY: {Number}

This property holds the vertical pixel position where the event occurred relative to the to the top-left corner of the display associated with the event.

This property may be undefined, if the event is not related to a display or a specific position, which is the case, for example, for keyboard events, map view change events or some customer defined events.

Interface: Properties

This interface is a member of [nokia.maps.map.Object](#).

here

Interface Summary

This interface defines the properties (keys) that can be passed to the constructor of all classes derived from *nokia.maps.map.Object*.

[For full details, see *nokia.maps.map.Object.Properties*]

Table 176: Property Summary

Properties
visibility: {Boolean} This property is a flag that determines the initial visibility of the created Object instance.
zIndex: {Number} This property holds the initial value of the z-index relative to the siblings in parent container.

Interface Description

This interface defines the properties (keys) that can be passed to the constructor of all classes derived from *nokia.maps.map.Object*.

Property Details

visibility: {Boolean}

This property is a flag that determines the initial visibility of the created Object instance. An object are visible if this property is `true` and if all its parent objects are visible (see also *nokia.maps.map.Object.isVisible()*).

zIndex: {Number}

This property holds the initial value of the z-index relative to the siblings in parent container. The z-index is used to specify the stacking order within a container. Objects with the highest values are placed on top (closer to the viewer). The overall stack order is defined hierarchically according to the z-index: objects are ordered within the container to which they belong, and containers are ordered within the *Display* instance. If a container A has a higher z-index than container B, objects that belong to A appear closer to the viewer than any object in B, even if all or some of the objects in B may have a higher z-index than some or all objects in A.

Class: Polygon

This class is a member of *nokia.maps.map*.

here

Extends: [nokia.maps.map.Polyline](#)

Class Summary

This class represent visible polygon on map.

[For full details, see [nokia.maps.map.Polygon](#)]

Table 177: Property Summary

Properties
brush : { nokia.maps.util.Brush } This property specifies the brush used to fill the shape of the polygon.

Directly Inherited Properties

Inherited from class [nokia.maps.map.Polyline](#) :

[arrows](#), [path](#), [pen](#)

Inherited from class [nokia.maps.map.Spatial](#) :

[simplify](#)

Inherited from class [nokia.maps.map.Object](#) :

[CHANGE_SPATIAL](#), [CHANGE_VISUAL](#), [CHANGE_ZINDEX](#), [visibility](#), [zIndex](#)

Inherited from class [nokia.maps.dom.EventTarget](#) :

[isDraggable](#), [isEventTarget](#), [parentNode](#), [parentNodes](#)

Inherited from class [nokia.maps.map.provider.IData](#) :

[id](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.Polyline](#) :

[getNearest](#), [getNearestIndex](#)

Inherited from class [nokia.maps.map.Object](#) :

[destroy](#), [getBoundingBox](#), [getDisplay](#), [getParent](#), [getProvider](#), [getRoot](#), [isVisible](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

here

Inherited from class *nokia.maps.dom.EventTarget* :

addListener, addListenerNS, addListeners, disableDrag, disableUserSelect, dispatch, enableDrag, enableUserSelect, hitTest, insertListener, insertListenerNS, removeAllListeners, removeListener, removeListenerNS

Inherited from class *nokia.maps.map.provider.IData* :

getInvalidations

Event Summary

Directly Inherited Events

Inherited from class *nokia.maps.dom.MouseEventTarget* :

click, dblclick, longpress, mousedown, mouseenter, mouseleave, mousemove, mouseout, mouseover, mouseup, mousewheel

Inherited from class *nokia.maps.dom.DragEventTarget* :

drag, dragend, dragenter, dragleave, dragover, dragstart, drop

Inherited from class *nokia.maps.dom.TouchEventTarget* :

dbltap, gesturechange, gestureend, gesturestart, longpress, tap, touchend, touchmove, touchstart

Class Description

This class represents map object that is, in effect, a closed polyline (*nokia.maps.map.Polyline*). In other words, the last point in the array of points that define the line is connected with the first point.

Constructor Details

nokia.maps.map.Polygon(path, props)

This method initializes an instance of Polygon. You can use it to set the points that define the polygon and set the brush.

Parameters:

path: {*nokia.maps.geo.Strip* | *nokia.maps.geo.Coordinate*[][]}

An object containing the points (geographical locations) that define the polygon; the object can be an instance of *nokia.maps.geo.Strip* or an array

here

of objects, each containing the latitude and longitude of a geographical location

props: {[nokia.maps.map.Polygon.Properties](#)} [optional]

An object that names polygon properties to initialize and provides their values

Property Details

brush: {[nokia.maps.util.Brush](#)}

This property specifies the brush used to fill the shape of the polygon. See [nokia.maps.util.Brush](#) for default values.

Interface: Properties

This interface is a member of [nokia.maps.map.Polygon](#).

Interface Summary

This interface defines the properties (keys) that can be passed to the constructor of [nokia.maps.map.Polygon](#).

[For full details, see [nokia.maps.map.Polygon.Properties](#)]

Table 178: Property Summary

Properties
brush: { nokia.maps.util.IBrush }
This property specifies the brush used to fill the shape of the polygon.

Directly Inherited Properties

Inherited from class [nokia.maps.map.Polyline.Properties](#) :

[arrows](#), [pen](#)

Inherited from class [nokia.maps.map.Spatial.Properties](#) :

[simplify](#)

Inherited from class [nokia.maps.map.Object.Properties](#) :

[visibility](#), [zIndex](#)

here

Interface Description

This interface defines the properties (keys) that can be passed to the constructor of [nokia.maps.map.Polygon](#).

Property Details

brush: {[nokia.maps.util.IBrush](#)}

This property specifies the brush used to fill the shape of the polygon.

Class: Polyline

This class is a member of [nokia.maps.map](#).

Class Summary

This class represents a visible line that connects a number of points on the map.

[For full details, see [nokia.maps.map.Polyline](#)]

Table 179: Property Summary

Properties
arrows: {nokia.maps.util.Arrows} This property holds an object that defines the attributes of arrows drawn recurrently on top of the polyline on the map to indicate the direction (for example the direction of travel along a route).
path: {nokia.maps.geo.Shape} This property holds a nokia.maps.geo.Shape object that contains the vertices for the polyline.
pen: {nokia.maps.util.Pen} This property holds a nokia.maps.util.Pen object that defines the drawing properties of the given Polyline (stroke color, line width, etc).

Directly Inherited Properties

Inherited from class [nokia.maps.map.Spatial](#) :

[simplify](#)

Inherited from class [nokia.maps.map.Object](#) :

[CHANGE_SPATIAL](#), [CHANGE_VISUAL](#), [CHANGE_ZINDEX](#), [visibility](#), [zIndex](#)

Inherited from class [nokia.maps.dom.EventTarget](#) :

[isDraggable](#), [isEventTarget](#), [parentNode](#), [parentNodes](#)

here

Inherited from class [nokia.maps.map.provider.IData](#) :

[id](#)

Table 180: Method Summary

Methods
getNearest (coord) : {nokia.maps.geo.Coordinate} This method retrieves the point in the given instance of PolyLine that is the closest to the location specified by the caller. getNearestIndex (coord) : {Number} This method obtains the index of a point that belongs to the given instance of PolyLine and lies closest to the location specified by the caller.

Directly Inherited Methods

Inherited from class [nokia.maps.map.Object](#) :

[destroy](#), [getBoundingBox](#), [getDisplay](#), [getParent](#), [getProvider](#), [getRoot](#), [isVisible](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Inherited from class [nokia.maps.dom.EventTarget](#) :

[addListener](#), [addListenerNS](#), [addListeners](#), [disableDrag](#), [disableUserSelect](#), [dispatch](#), [enableDrag](#), [enableUserSelect](#), [hitTest](#), [insertListener](#), [insertListenerNS](#), [removeAllListeners](#), [removeListener](#), [removeListenerNS](#)

Inherited from class [nokia.maps.map.provider.IData](#) :

[getInvalidations](#)

Event Summary

Directly Inherited Events

Inherited from class [nokia.maps.dom.MouseEventTarget](#) :

[click](#), [dblclick](#), [longpress](#), [mousedown](#), [mouseenter](#), [mouseleave](#), [mousemove](#), [mouseout](#), [mouseover](#), [mouseup](#), [mousewheel](#)

Inherited from class [nokia.maps.dom.DragEventTarget](#) :

[drag](#), [dragend](#), [dragenter](#), [dragleave](#), [dragover](#), [dragstart](#), [drop](#)

Inherited from class [nokia.maps.dom.TouchEventTarget](#) :

[dbltap](#), [gesturechange](#), [gestureend](#), [gesturestart](#), [longpress](#), [tap](#), [touchend](#), [touchmove](#), [touchstart](#)

here

Class Description

`PolyLine` represents a visible line that connects two or more points on the map. The section that connects any two consecutive points is always straight and therefore indicates the shortest distance between those points.

Constructor Details

`nokia.maps.map.PolyLine(coords, props)`

This method creates and initializes an instance of `PolyLine`.

Parameters:

`coords`: `{nokia.maps.geo.Strip | nokia.maps.geo.Shape | nokia.maps.geo.Coordinate[]}`

An object containing the points (geographical locations) that define the polyline; the object can be an instance of `nokia.maps.geo.Strip`, a `nokia.maps.geo.Shape` or an array of objects, each containing the latitude and longitude of a geographical location.

`props`: `{nokia.maps.map.PolyLine.Properties}` [optional]

An object that names `PolyLine` properties and their initial values.

Property Details

`arrows`: `{nokia.maps.util.Arrows}`

This property holds an object that defines the attributes of arrows drawn recurrently on top of the polyline on the map to indicate the direction (for example the direction of travel along a route). If the property is not specified or if it is set to an invalid value, no arrows are drawn.

`path`: `{nokia.maps.geo.Shape}`

This property holds a `nokia.maps.geo.Shape` object that contains the vertices for the polyline.

`pen`: `{nokia.maps.util.Pen}`

This property holds a `nokia.maps.util.Pen` object that defines the drawing properties of the given `PolyLine` (stroke color, line width, etc.) See [nokia.maps.util.Pen](#) for default values.

here

Method Details

getNearest(coord): {[nokia.maps.geo.Coordinate](#)}

This method retrieves the point in the given instance of `Polyline` that is the closest to the location specified by the caller.

Parameters:

coord: {[nokia.maps.geo.Coordinate](#)}

An object containing the coordinates of a location for which to find the closest point on the polyline.

Returns:

{[nokia.maps.geo.Coordinate](#)}

An instance of `nokia.maps.geo.Coordinate` containing the latitude and longitude of a point on the polyline.

getNearestIndex(coord): {Number}

This method obtains the index of a point that belongs to the given instance of `Polyline` and lies closest to the location specified by the caller.

Parameters:

coord: {[nokia.maps.geo.Coordinate](#)}

An object containing the coordinates of a location.

Returns:

{Number}

A number representing the index of the point on the polyline that is closest to the location provided by the caller.

Interface: Properties

This interface is a member of [nokia.maps.map.Polyline](#).

here

Interface Summary

This interface defines the properties (keys) that can be passed to the constructor of [nokia.maps.map.Polyline](#).

[For full details, see [nokia.maps.map.Polyline.Properties](#)]

Table 181: Property Summary

Properties
arrows: {nokia.maps.util.Arrows} This property holds an object that defines the attributes of arrows drawn recurrently on top of the polyline on the map to indicate the direction (for example the direction of travel along a route).
pen: {nokia.maps.util.IPen} This property holds a Pen object that defines the drawing properties of the given Polyline (stroke color, line width, etc).

Directly Inherited Properties

Inherited from class [nokia.maps.map.Spatial.Properties](#) :

[simplify](#)

Inherited from class [nokia.maps.map.Object.Properties](#) :

[visibility](#), [zIndex](#)

Interface Description

This interface defines the properties (keys) that can be passed to the constructor of [nokia.maps.map.Polyline](#).

Property Details

arrows: {[nokia.maps.util.Arrows](#)}

This property holds an object that defines the attributes of arrows drawn recurrently on top of the polyline on the map to indicate the direction (for example the direction of travel along a route). If the property is not specified or if it is set to an invalid value, no arrows are drawn.

pen: {[nokia.maps.util.IPen](#)}

This property holds a Pen object that defines the drawing properties of the given Polyline (stroke color, line width, etc.)

here

Class: Rectangle

This class is a member of [nokia.maps.map](#).

Class Summary

This class represents visible rectangular area on map.

[For full details, see [nokia.maps.map.Rectangle](#)]

Table 182: Property Summary

Properties
boundingBox : { nokia.maps.geo.BoundingBox }
The bounds of the rectangle.
simplify : {Number}
This property overrides nokia.maps.map.Spatial#simplify .

Directly Inherited Properties

Inherited from class [nokia.maps.map.Polygon](#) :

[brush](#)

Inherited from class [nokia.maps.map.Polyline](#) :

[arrows](#), [path](#), [pen](#)

Inherited from class [nokia.maps.map.Spatial](#) :

[simplify](#)

Inherited from class [nokia.maps.map.Object](#) :

[CHANGE_SPATIAL](#), [CHANGE_VISUAL](#), [CHANGE_ZINDEX](#), [visibility](#), [zIndex](#)

Inherited from class [nokia.maps.dom.EventTarget](#) :

[isDraggable](#), [isEventTarget](#), [parentNode](#), [parentNodes](#)

Inherited from class [nokia.maps.map.provider.IData](#) :

[id](#)

Method Summary

Directly Inherited Methods

here

Inherited from class *nokia.maps.map.Polyline* :

getNearest, getNearestIndex

Inherited from class *nokia.maps.map.Object* :

destroy, getBoundingBox, getDisplay, getParent, getProvider, getRoot, isVisible

Inherited from class *nokia.maps.util.OObject* :

addObserver, get, remove, removeObserver, set

Inherited from class *nokia.maps.dom.EventTarget* :

addListener, addListenerNS, addListeners, disableDrag, disableUserSelect, dispatch, enableDrag, enableUserSelect, hitTest, insertListener, insertListenerNS, removeAllListeners, removeListener, removeListenerNS

Inherited from class *nokia.maps.map.provider.IData* :

getInvalidations

Event Summary

Directly Inherited Events

Inherited from class *nokia.maps.dom.MouseEventTarget* :

click, dblclick, longpress, mousedown, mouseenter, mouseleave, mousemove, mouseout, mouseover, mouseup, mousewheel

Inherited from class *nokia.maps.dom.DragEventTarget* :

drag, dragend, dragenter, dragleave, dragover, dragstart, drop

Inherited from class *nokia.maps.dom.TouchEventTarget* :

dbltap, gesturechange, gestureend, gesturestart, longpress, tap, touchend, touchmove, touchstart

Class Description

The class defines a map object with a rectangular shape.

Constructor Details

nokia.maps.map.Rectangle(bounds, props)

This method initializes an instance of *nokia.maps.map.Rectangle* setting the bounding box for the object and other properties specified by the caller.

here

Parameters:

- bounds:** {[nokia.maps.geo.BoundingBox](#)}
- A bounding box object that determines the location of the rectangle and its size
- props:** {[nokia.maps.map.Rectangle.Properties](#)} [optional]
- An object that names the properties of the rectangle to initialize and provides their values

Property Details

boundingBox: {[nokia.maps.geo.BoundingBox](#)}

The bounds of the rectangle.

simplify: {Number}

This property overrides [nokia.maps.map.Spatial#simplify](#). The value of this property is 0 by default, because it is not visually nice to simplify rectangles. It is also not necessary, because Rectangle always consists of 4 points.

Default Value: 0

Interface: Properties

This interface is a member of [nokia.maps.map.Rectangle](#).

Interface Summary

This interface defines the properties (keys) that can be passed to the constructor for [nokia.maps.map.Rectangle](#).

[For full details, see [nokia.maps.map.Rectangle.Properties](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.Polygon.Properties](#) :

[brush](#)

here

Inherited from class [nokia.maps.map.Polyline.Properties](#) :

[arrows](#), [pen](#)

Inherited from class [nokia.maps.map.Spatial.Properties](#) :

[simplify](#)

Inherited from class [nokia.maps.map.Object.Properties](#) :

[visibility](#), [zIndex](#)

Interface Description

This interface defines the properties (keys) that can be passed to the constructor for [nokia.maps.map.Rectangle](#).

Class: Spatial

This class is a member of [nokia.maps.map](#).

Extends: [nokia.maps.map.Object](#)

Class Summary

This class is a base class for all geo spatial map objects.

[For full details, see [nokia.maps.map.Spatial](#)]

Table 183: Property Summary

Properties
simplify: {Number} The value of this property is specified as a number of pixels and indicates if and how much the rendering result is to be simplified.

Directly Inherited Properties

Inherited from class [nokia.maps.map.Object](#) :

[CHANGE_SPATIAL](#), [CHANGE_VISUAL](#), [CHANGE_ZINDEX](#), [visibility](#), [zIndex](#)

Inherited from class [nokia.maps.dom.EventTarget](#) :

[isDraggable](#), [isEventTarget](#), [parentNode](#), [parentNodes](#)

Inherited from class [nokia.maps.map.provider.IData](#) :

[id](#)

here

Method Summary

Directly Inherited Methods

Inherited from class *nokia.maps.map.Object* :

destroy, getBoundingBox, getDisplay, getParent, getProvider, getRoot, isVisible

Inherited from class *nokia.maps.util.OObject* :

addObserver, get, remove, removeObserver, set

Inherited from class *nokia.maps.dom.EventTarget* :

addListener, addListenerNS, addListeners, disableDrag, disableUserSelect, dispatch, enableDrag, enableUserSelect, hitTest, insertListener, insertListenerNS, removeAllListeners, removeListener, removeListenerNS

Inherited from class *nokia.maps.map.provider.IData* :

getInvalidations

Event Summary

Directly Inherited Events

Inherited from class *nokia.maps.dom.MouseEventTarget* :

click, dblclick, longpress, mousedown, mouseenter, mouseleave, mousemove, mouseout, mouseover, mouseup, mousewheel

Inherited from class *nokia.maps.dom.DragEventTarget* :

drag, dragend, dragenter, dragleave, dragover, dragstart, drop

Inherited from class *nokia.maps.dom.TouchEventTarget* :

dbltap, gesturechange, gestureend, gesturestart, longpress, tap, touchend, touchmove, touchstart

Class Description

This is an abstract base class for all objects that have a geo-spatial dimension.

Constructor Details

nokia.maps.map.Spatial(props)

This method initializes an object based on this class.

Parameters:

here

props: `{nokia.maps.map.Object.Properties}`

An object specifying the initial values of the properties defined on this class

Property Details

simplify: {Number}

The value of this property is specified as a number of pixels and indicates if and how much the rendering result is to be simplified. If the value is 0, no simplification is applied. Otherwise, the complexity of the rendering result is reduced so that points within a distance smaller than the number of pixels indicated by this property are ignored.

Default Value: 3

Interface: *Properties*

This interface is a member of [nokia.maps.map.Spatial](#).

Interface Summary

This interface defines the properties (keys) that can be passed to the constructor of all classes derived from [nokia.maps.map.Spatial](#).

[For full details, see [nokia.maps.map.Spatial.Properties](#)]

Table 184: Property Summary

Properties
simplify: {Number} The value of this property is specified as a number of pixels and indicates if and how much the rendering result is to be simplified.

Directly Inherited Properties

Inherited from class [nokia.maps.map.Object.Properties](#) :

[visibility](#), [zIndex](#)

Interface Description

This interface defines the properties (keys) that can be passed to the constructor of all classes derived from [nokia.maps.map.Spatial](#).

here

Property Details

simplify: {Number}

The value of this property is specified as a number of pixels and indicates if and how much the rendering result is to be simplified. If the value is 0, no simplification is applied. Otherwise, the complexity of the rendering result is reduced so that points within a distance smaller than the number of pixels indicated by this property are ignored.

Class: StandardMarker

This class is a member of [nokia.maps.map](#).

Extends: [nokia.maps.map.Marker](#)

Class Summary

This class represents a marker with standardized icon.

[For full details, see [nokia.maps.map.StandardMarker](#)]

Table 185: Property Summary

Properties
brush: {nokia.maps.util.Brush} This property holds the nokia.maps.util.Brush object which defines color of the interior of the shape in which the marker's text label is rendered.
pen: {nokia.maps.util.Pen} This property holds the nokia.maps.util.Pen object which defines color of the outline of the shape in which the marker's text label is rendered.
shape: {String} This property specifies the shape that is used to display the marker label.
text: {String} This property holds the text (label) rendered in the center of the shape.
textPen: {nokia.maps.util.Pen} This property holds the nokia.maps.util.Pen object which defines color of the text in the marker's label.

Directly Inherited Properties

Inherited from class [nokia.maps.map.Marker](#) :

[anchor](#), [coordinate](#), [hitArea](#), [icon](#)

Inherited from class [nokia.maps.map.Object](#) :

here

CHANGE_SPATIAL, CHANGE_VISUAL, CHANGE_ZINDEX, visibility, zIndex

Inherited from class *nokia.maps.dom.EventTarget* :

isDraggable, isEventTarget, parentNode, parentNodes

Inherited from class *nokia.maps.map.provider.IData* :

id

Method Summary

Directly Inherited Methods

Inherited from class *nokia.maps.map.Marker* :

getDisplayBoundingBox, getDisplayOffset, getIconForRendering, hitTest

Inherited from class *nokia.maps.map.Object* :

destroy, getBoundingBox, getDisplay, getParent, getProvider, getRoot, isVisible

Inherited from class *nokia.maps.util.OObject* :

addObserver, get, remove, removeObserver, set

Inherited from class *nokia.maps.dom.EventTarget* :

addListener, addListenerNS, addListeners, disableDrag, disableUserSelect, dispatch, enableDrag, enableUserSelect, hitTest, insertListener, insertListenerNS, removeAllListeners, removeListener, removeListenerNS

Inherited from class *nokia.maps.map.provider.IData* :

getInvalidations

Event Summary

Directly Inherited Events

Inherited from class *nokia.maps.dom.MouseEventTarget* :

click, dblclick, longpress, mousedown, mouseenter, mouseleave, mousemove, mouseout, mouseover, mouseup, mousewheel

Inherited from class *nokia.maps.dom.DragEventTarget* :

drag, dragend, dragenter, dragleave, dragover, dragstart, drop

Inherited from class *nokia.maps.dom.TouchEventTarget* :

dbltap, gesturechange, gestureend, gesturestart, longpress, tap, touchend, touchmove, touchstart

here

Class Description

A `StandardMarker` represents a location on the map and visibly identifies that location with an icon. It is very similar in terms of properties and supported functionality to [nokia.maps.map.Marker](#), but in addition, it supports a text label and various text properties for it.

Constructor Details

nokia.maps.map.StandardMarker(coord, props)

This method creates an instance of `StandardMarker`.

Parameters:

coord:	{nokia.maps.geo.Coordinate}
	An object containing the geographical coordinates of the marker
props:	{nokia.maps.map.StandardMarker.Properties} [optional]
	The initial values of marker properties

Property Details

brush: [{nokia.maps.util.Brush}](#)

This property holds the [nokia.maps.util.Brush](#) object which defines color of the interior of the shape in which the marker's text label is rendered. If the property is not set, the original color defined in the specific shape type is used. The default color for "balloon" is #1080DD.

pen: [{nokia.maps.util.Pen}](#)

This property holds the [nokia.maps.util.Pen](#) object which defines color of the outline of the shape in which the marker's text label is rendered. Default stroke color is set to #FFF.

shape: {String}

This property specifies the shape that is used to display the marker label. The following strings can be used right now to set its value:

- "balloon"

There will be more coming soon.

Default Value: "balloon"

here

text: {String}

This property holds the text (label) rendered in the center of the shape. If the rendered text is longer than the screen bounding box of the shape, the text is cropped.

Default Value: ""

textPen: {[nokia.maps.util.Pen](#)}

This property holds the [nokia.maps.util.Pen](#) object which defines color of the text in the marker's label. Default stroke color is #FFF.

Interface: Properties

This interface is a member of [nokia.maps.map.StandardMarker](#).

Interface Summary

This interface defines the properties (keys) that can be passed to the [nokia.maps.map.StandardMarker](#) constructor.

[For full details, see [nokia.maps.map.StandardMarker.Properties](#)]

Table 186: Property Summary

Properties
brush: {nokia.maps.util.IBrush} This property define the nokia.maps.util.IBrush object which color property is used for the interior of the shape in which the marker's text label is rendered.
pen: {nokia.maps.util.IPen} This property define the nokia.maps.util.IPen object which stroke color is used for the outline of the shape in which the marker's text label is rendered.
shape: {String} This property specifies the shape that is used to display the marker label.
style: {Object} The property specifies the css style which will be applied to zoom rectangle.
text: {String} This property define the text (label) rendered in the center of the shape.
textPen: {nokia.maps.util.IPen}

here

Properties

This property define the [nokia.maps.util.IPen](#) object which stroke color is used for the text in the marker's label.

Directly Inherited Properties

Inherited from class [nokia.maps.map.Marker.Properties](#) :

[anchor](#), [draggable](#), [hitArea](#), [icon](#)

Inherited from class [nokia.maps.map.Object.Properties](#) :

[visibility](#), [zIndex](#)

Interface Description

This interface defines the properties (keys) that can be passed to the [nokia.maps.map.StandardMarker](#) constructor.

Property Details

brush: {[nokia.maps.util.IBrush](#)}

This property define the [nokia.maps.util.IBrush](#) object which color property is used for the interior of the shape in which the marker's text label is rendered. If the property is not set, the original color defined in the specific shape type is used. The default color for "balloon" is #1080DD.

pen: {[nokia.maps.util.IPen](#)}

This property define the [nokia.maps.util.IPen](#) object which stroke color is used for the outline of the shape in which the marker's text label is rendered. Default stroke color is set to #FFF.

shape: {String}

This property specifies the shape that is used to display the marker label. The following strings can be used right now to set its value:

- "balloon"

There will be more coming soon.

style: {Object}

here

The property specifies the css style which will be applied to zoom rectangle. Allowed style properties are:

- border
- background
- opacity

Example:

```
border: "1px solid #F00",
        background: "#0F0",
        opacity: 0.5
    }
```

text: {String}

This property define the text (label) rendered in the center of the shape. If the rendered text is longer than the screen bounding box of the shape, the text is cropped.

textPen: {[nokia.maps.util.IPen](#)}

This property define the [nokia.maps.util.IPen](#) object which stroke color is used for the text in the marker's label. Default stroke color is #FFF.

Namespace: component

This namespace is a member of [nokia.maps.map](#).

Namespace Summary

This namespace provides UI components that can be added to a map display and others that support various types of interactive behavior such as right-mouse click, draggability, mouse gestures, etc.

Namespace Description

This namespace provides UI components that can be added to a map display and others that support various types of interactive behavior such as right-mouse click, draggability, mouse gestures, etc.

Class: Behavior

This class is a member of [nokia.maps.map.component](#).

here

Extends: [nokia.maps.map.component.Component](#)

Class Summary

This is a meta component that adds common user interface elements to the map.

[For full details, see [nokia.maps.map.component.Behavior](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

[mapDisplay](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :

[attach](#), [destroy](#), [detach](#), [getId](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

Behavior is a collection of user interface elements cumulatively responsible for supporting direct user-interaction with the map. Behavior adds or removes the following map user interface components:

- [nokia.maps.map.component.zoom.MouseWheel](#)
- [nokia.maps.map.component.zoom.DoubleClick](#)
- [nokia.maps.map.component.zoom.DoubleTap](#)
- [nokia.maps.map.component.zoom.Gesture](#)
- [nokia.maps.map.component.panning.Drag](#)
- [nokia.maps.map.component.panning.Kinetic](#)
- [nokia.maps.map.component.objects.DragMarker](#)

Constructor Details

`nokia.maps.map.component.Behavior()`

here

This method initializes an instance of Behavior.

Class: Component

This class is a member of [nokia.maps.map.component](#).

Extends: [nokia.maps.util.OObject](#)

Class Summary

This class is a base class for all map components.

[For full details, see [nokia.maps.map.component.Component](#)]

Table 187: Property Summary

Properties
mapDisplay : { nokia.maps.map.Display }
This property holds a reference to the map . Display instance to which the given component belongs.

Table 188: Method Summary

Methods
attach (mapDisplay)
This method will be invoked when the component is attached to a map and is not meant to be called directly.
destroy ()
This method is called from Display when Display is destroyed.
detach (mapDisplay)
This method will be invoked when the component is detached from a map and is not meant to be called directly
getId () : {String}
This method retrieves the unique identifier of the component.

Directly Inherited Methods

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

Component is an abstract class that serves as basis for defining replaceable software modules that extend the basic map with additional functionality. Components allow API users to:

here

- include new functionality
- replace existing functionality with a different implementation
- deactivate existing functionality.

A component may fire custom events.

The `Component` class is implemented in several packages outside the `map` package, for example in routing and search.

Reference counting

The fact that components are replaceable and that they can add or remove other components, creating and potentially disrupting inter-component dependencies, makes a reference counting mechanism necessary. The basic principle is that whenever a component is created its reference count is incremented to indicate that the creator requires it (and holds a reference to it). The count is also incremented when a component already exists, but a new attempt to create it is made - thus, the count goes up, but duplicate components with the same id are not created. An attempt to delete/remove a component, causes its reference count to be decremented.

This mechanism prevents not only component duplication, but also the removal of components while other components dependent on them for as long as the dependencies exist. Caution when physically removing components is necessary, because on removal, a component's detach callback is called to delete its reference to the `map Display`. As a result, the component becomes unusable in an invalid state.

However, this mechanism remains transparent and you, as the API user simply need to call the methods `addComponent()` and `removeComponent()` of the `map Display` class to add and remove components, respectively. Here is an example:

```
// If a Kinetic component already exists, addComponent()  
// returns it instead of creating a duplicate.  
var component = display.addComponent(new  
    nokia.maps.map.component.panning.Kinetic());  
  
// If any other component or application part holds a reference  
// to the component, this call does not remove the component, but  
// decrements the reference counter by one, otherwise it removes  
// the component (if the reference counter reached zero).  
display.removeComponent(component);  
  
// Another possibility to remove a component is:  
display.removeComponent( display.getComponentById("panning.Kinetic") );
```

Property Details

mapDisplay: {*nokia.maps.map.Display*}

here

This property holds a reference to the `map.Display` instance to which the given component belongs.

Method Details

attach(mapDisplay)

This method will be invoked when the component is attached to a map and is not meant to be called directly.

Note that a component must not be bound to two instances of `map.Display` at the same time!

Parameters:

mapDisplay: {*nokia.maps.map.Display*}

A reference to the `map.Display` instance to which the given component is bound.

destroy()

This method is called from `Display` when `Display` is destroyed. It is also intended that you call it when you detach component from `Display` and do not want to use component instance anymore. It will free all allocated resources.

detach(mapDisplay)

This method will be invoked when the component is detached from a map and is not meant to be called directly

Parameters:

mapDisplay: {*nokia.maps.map.Display*}

A reference to the `map.Display` instance from which this component is to be removed.

getId(): {String}

This method retrieves the unique identifier of the component. The id is guaranteed to be unique, because it is not possible to add two components with the same id to the map.

here

Returns:

{String} The component identifier

Class: *ContextMenu*

This class is a member of [nokia.maps.map.component](#).

Extends: [nokia.maps.map.component.Component](#)

Class Summary

This class represents a context menu that is displayed on right-click with the mouse or long-tap in the map.

[For full details, see [nokia.maps.map.component.ContextMenu](#)]

Table 189: Property Summary

Properties
static captureHandler : This static method is the default implementation for "Capture" context menu entry.
static mapFeedbackHandler : This static method is the default implementation for "Report a problem" context menu entry.
static routingHandler : This static method is the default implementation of "Routing" context menu section.
static zoomHandler : This static method is the default implementation that provides access to map zoom functionality from the context menu.

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

[mapDisplay](#)

Table 190: Method Summary

Methods
addHandler (handler) This method adds a user-defined handler to the context menu.
removeHandler (handler) This method removes a previously added handler from the context menu.

here

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :

[attach](#), [destroy](#), [detach](#), [getId](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This class represents a context menu that is displayed on right-click with the mouse or long-tap in the map. @class The context menu displays a context menu on the map. By default the context menu contains two entries that support zooming the map in and out one level at a time.

The context menu may be extended via handlers. A handler is responsible for a group of menu entries and you can add new entries to the group as required. Each entry consists of a label and an associated callback that implements an action (functionality) specific to the menu entry.

Constructor Details

nokia.maps.map.component.ContextMenu()

This method initializes a new instance of ContextMenu

Example:

```
var myHandler = function(contextMenuEvent, group) {
    group.addEntry(
        "pan left",
        function(activationEvent) {
            this.mapDisplay.pan(0, 0, -200, 0);
        });
}

var contextMenu = new nokia.maps.map.component.ContextMenu();
contextMenu.addHandler(myHandler);

display.components.add(contextMenu);
```

Property Details

static **captureHandler** :

This static method is the default implementation for "Capture" context menu entry.

It is NOT added to the ContextMenu by default and can be manually added by calling [nokia.maps.map.component.ContextMenu#addHandler](#) with this handler as a parameter. This handler

here

is only supported in certain browsers (Mozilla Firefox and Internet Explorer 9 and 10). In other environments, adding this handler will not have any effect.

static mapFeedbackHandler:

This static method is the default implementation for "Report a problem" context menu entry. It is added to the ContextMenu at construction time and lazy loads widget when menu is shown for the first time.

static routingHandler:

This static method is the default implementation of "Routing" context menu section. It is added to the ContextMenu at construction time IF routing was loaded.

static zoomHandler:

This static method is the default implementation that provides access to map zoom functionality from the context menu. ([nokia.maps.map.component.ContextMenu.Group#addEntry](#)) The method can be added as a handler (using [nokia.maps.map.component.ContextMenu#addHandler](#)) in order to activate the functionality in the context menu.

Method Details

addHandler(handler)

This method adds a user-defined handler to the context menu. The handler callback is called with the event that activated the context menu and a group that is unique to the handler function. To add group entries, invoke the method [addEntry\(\)](#) ([nokia.maps.map.component.ContextMenu.Group#addEntry](#)).

Parameters:

handler: {Function}

removeHandler(handler)

This method removes a previously added handler from the context menu.

Parameters:

handler: {Function}

here

A reference to the handler to remove

Class: Group

This class is a member of [nokia.maps.map.component.ContextMenu](#).

Class Summary

This class represents group of entries within the context menu.

[For full details, see [nokia.maps.map.component.ContextMenu.Group](#)]

Table 191: Method Summary

Methods
addEntry (label, callback) This method adds an entry to the group.

Class Description

The class `ContextMenu.Group` represents an abstract group of entries within the context menu. Each handler function that is registered on the context menu is associated with a specific group object to which it can add entries.

If a handler does not to add entries to its group, it is not included in the context menu.

Constructor Details

`nokia.maps.map.component.ContextMenu.Group()`

This method initializes a new instance of `ContextMenu.Group`. This constructor is invoked by the context menu and a fully initialized Group instance is passed to each handler registered on the context menu.

Method Details

`addEntry`(label, callback)

This method adds an entry to the group. An entry consists of a label and an optional callback function to be invoked upon activation of the entry. The label can be a String (for a static label) or a callback function which receives the corresponding DOM node.

Parameters:

here

label:	{String Function}
	The label of the entry, or a callback function to be invoked when the node for this entry is created
callback:	{Function} [optional]
	A callback function to be invoked when the entry is activated

Example:

```
//adding an entry with a static text label and an activation callback
group.addEntry("my label", function(activationEvent) {
    //activation call back code
});
//adding an entry with a label callback and an activation callback
group.addEntry(
    function(domNode) {
        //add label dynamically
    },
    function(activationEvent) {
        //activation call back code
    }
);
```

Class: DistanceMeasurement

This class is a member of [nokia.maps.map.component](#).

Extends: [nokia.maps.map.component.Component](#)

Class Summary

This component helps calculate distances between geographical locations indicated by user clicks.

[For full details, see [nokia.maps.map.component.DistanceMeasurement](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

[mapDisplay](#)

here

Table 192: Method Summary

Methods
<code>formatLength</code> (length) : {Number String} This method obtains the formatted distance that can be use in distance labels.

Directly Inherited Methods

Inherited from class `nokia.maps.map.component.Component` :

`attach`, `destroy`, `detach`, `getId`

Inherited from class `nokia.maps.util.OObject` :

`addObserver`, `get`, `remove`, `removeObserver`, `set`

Class Description

DistanceMeasurement adds direct user-interaction to the map. The user can click on the map to specify locations and build a path whose length is measured. The locations are indicated by markers and the path by lines connecting the markers.

A marker that has been added to the map in this way can be removed by holding down the CTRL key and clicking on the marker. The user can also split the line connecting two markers and create a new marker by dragging the line or clicking on the line between markers

Constructor Details

`nokia.maps.map.component.DistanceMeasurement`(props)

This method creates a DistanceMeasurement tool component which can be added to a map display.

Parameters:

props: `{nokia.maps.map.component.DistanceMeasurement.Properties}`
[optional]

An object which can contain properties and/or methods which should be appended to instances of this class; the object can be used to extend the component.

Method Details

`formatLength`(length): {Number | String}

here

This method obtains the formatted distance that can be use in distance labels. (The method is called to format distance labels). This method should be rewritten in derived classes, for example, if there is a requirement to round numbers and to use different system of measurement.

Parameters:

length: {Number}
A value indicating the distance in meters

Returns:

{Number | String} A numeric value or string representing the formatted distance

Interface: Properties

This interface is a member of [nokia.maps.map.component.DistanceMeasurement](#).

Interface Summary

This interface defines the properties (keys) that can be passed to the map [nokia.maps.map.component.DistanceMeasurement](#) constructor.

[For full details, see [nokia.maps.map.component.DistanceMeasurement.Properties](#)]

Table 193: Property Summary

Properties
endMarkerOptions : { nokia.maps.map.Marker.Properties } This property holds options like icon or anchor for last marker.
hoverPen : { nokia.maps.util.IPen } This property holds pen for polyline connecting two markers which is highlighted when mouse pointer is above this line.
middleMarkerOptions : { nokia.maps.map.Marker.Properties } This property holds options like icon or anchor for middle markers.
normalPen : { nokia.maps.util.IPen } This property holds pen for polyline connecting two markers.
startMarkerOptions : { nokia.maps.map.Marker.Properties } This property holds options like icon or anchor for first marker.
textLabelOnFirstMarker : {Boolean} This property defines if text label is also displayed for first marker (true) or not (false).

here

Interface Description

This interface defines the properties (keys) that can be passed to the map [*nokia.maps.map.component.DistanceMeasurement*](#) constructor.

Property Details

endMarkerOptions: {[*nokia.maps.map.Marker.Properties*](#)}

This property holds options like icon or anchor for last marker.

hoverPen: {[*nokia.maps.util.IPen*](#)}

This property holds pen for polyline connecting two markers which is highlighted when mouse pointer is above this line.

middleMarkerOptions: {[*nokia.maps.map.Marker.Properties*](#)}

This property holds options like icon or anchor for middle markers.

normalPen: {[*nokia.maps.util.IPen*](#)}

This property holds pen for polyline connecting two markers.

startMarkerOptions: {[*nokia.maps.map.Marker.Properties*](#)}

This property holds options like icon or anchor for first marker.

textLabelOnFirstMarker: {Boolean}

This property defines if text label is also displayed for first marker (true) or not (false).

Class: InfoBubbles

This class is a member of [*nokia.maps.map.component*](#).

Extends: [*nokia.maps.map.component.Component*](#)

Class Summary

This component manages visibility of info bubbles on the map.

here

[For full details, see [nokia.maps.map.component.InfoBubbles](#)]

Table 194: Property Summary

Properties
readonly ALIGNMENT_ABOVE : {String} This identifier specifies a vertical alignment above the bubble's anchor.
readonly ALIGNMENT_BELOW : {String} This identifier specifies a vertical alignment below the bubble's anchor.
readonly ALIGNMENT_LEFT : {String} This identifier specifies a horizontal alignment to the left of the bubble's anchor.
readonly ALIGNMENT_RIGHT : {String} This identifier specifies a horizontal alignment to the right of the bubble's anchor.
readonly DIMENSION_AUTO : {String} This identifier specifies a width or height which is automatically determined.
readonly openBubbleHandles : { nokia.maps.util.OList } This property holds an observable list of handles to open bubbles.
options : { nokia.maps.map.component.InfoBubbles.Options } This property holds an observable object that allows for modification of the presentation options on all newly created bubbles that are displayed via the given component.

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

[mapDisplay](#)

Table 195: Method Summary

Methods
closeBubble (bubble) This method closes the bubble that is passed as a parameter.
initBubble (onUserClose, hideCloseButton) : { nokia.maps.map.component.InfoBubbles.Bubble } This method creates a new empty info bubble.
openBubble (content, coordinate, onUserClose, hideCloseButton) : { nokia.maps.map.component.InfoBubbles.Bubble } This method creates a new info bubble, updates it with content and shows it on the map.

Directly Inherited Methods

here

Inherited from class [nokia.maps.map.component.Component](#) :

[attach](#), [destroy](#), [detach](#), [getId](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

If added to the map, this component manages info bubbles. The component is responsible for adding and removing info bubbles. Each info bubble must have an instance of [nokia.maps.geo.Coordinate](#) and its contents is a string object.

This info bubbles components supports single or multiple info bubbles (see [nokia.maps.map.component.InfoBubbles#options](#)).

Constructor Details

nokia.maps.map.component.InfoBubbles()

This method initializes an instance of `nokia.maps.map.component.InfoBubbles` which can be added to a map Display.

Property Details

readonly **ALIGNMENT_ABOVE**: {String}

This identifier specifies a vertical alignment above the bubble's anchor.

readonly **ALIGNMENT_BELOW**: {String}

This identifier specifies a vertical alignment below the bubble's anchor.

readonly **ALIGNMENT_LEFT**: {String}

This identifier specifies a horizontal alignment to the left of the bubble's anchor.

readonly **ALIGNMENT_RIGHT**: {String}

This identifier specifies a horizontal alignment to the right of the bubble's anchor.

here

readonly **DIMENSION_AUTO**: {String}

This identifier specifies a width or height which is automatically determined.

readonly **openBubbleHandles**: {*nokia.maps.util.OList*}

This property holds an observable list of handles to open bubbles. When a bubble is shown, the respective handle is added to this list. When a bubble is closed, its handle is removed from this list.

options: {*nokia.maps.map.component.InfoBubbles.Options*}

This property holds an observable object that allows for modification of the presentation options on all newly created bubbles that are displayed via the given component.

Use *nokia.maps.util.OObject.set()* to set properties on this object to make sure changes are propagated correctly to the system.

Example:

```
var infoBubbles = new nokia.maps.map.component.InfoBubbles();
//map is an instance of nokia.maps.map.Display
map.components.add(infoBubbles);

// causes all subsequently created bubbles to open left of their anchor
// (unless there's not enough space on that side)
infoBubbles.options.set("defaultXAlignment", infoBubbles.ALIGNMENT_LEFT);
infoBubbles.openBubble("myInfoBubble", map.center);
```

Method Details

closeBubble(bubble)

This method closes the bubble that is passed as a parameter.

Parameters:

bubble: {*nokia.maps.map.component.InfoBubbles.Bubble*}

A handle to the info bubble

initBubble(onUserClose, hideCloseButton):
{*nokia.maps.map.component.InfoBubbles.Bubble*}

This method creates a new empty info bubble.

here

Note that you should use [nokia.maps.map.component.InfoBubbles#openBubble](#) if you do not want to [nokia.maps.map.component.InfoBubbles.Bubble#update](#) and [nokia.maps.map.component.InfoBubbles.Bubble#open](#) it by manually. This method is useful if you want to preload some content into bubble before showing it.

Parameters:

onUserClose: {Function} [optional]

A callback method which is called when user closes the bubble (by clicking on close button) to be placed.

hideCloseButton: {Boolean} [optional]

Hides close button if set to true.

Returns:

{[nokia.maps.map.component.InfoBubbles.Bubble](#)}

The handle of this bubble

openBubble(content, coordinate, onUserClose, hideCloseButton):
{[nokia.maps.map.component.InfoBubbles.Bubble](#)}

This method creates a new info bubble, updates it with content and shows it on the map.

Parameters:

content: {String | DomElement}

The content to be shown in the info bubble; it can be an HTML string, please note that Flash content in the bubble overlap other elements in the document

coordinate: {[nokia.maps.geo.Coordinate](#)}

An object containing the geographic coordinates of the location, where the bubble's anchor is to be placed.

onUserClose: {Function} [optional]

A callback method which is called when user closes the bubble (by clicking on close button) to be placed.

here

`hideCloseButton:` {Boolean} [optional]

Hides close button if set to true.

Returns:

{[nokia.maps.map.component.InfoBubbles.Bubble](#)}

The handle of this bubble

Interface: Bubble

This interface is a member of [nokia.maps.map.component.InfoBubbles](#).

Interface Summary

This interface defines a bubble which is shown on the map by an instance of [nokia.maps.map.component.InfoBubbles](#).

[For full details, see [nokia.maps.map.component.InfoBubbles.Bubble](#)]

Table 196: Property Summary

Properties
contentNode: {DomElement} This property holds the reference to the DOM node with the content of the bubble.
coordinate: {nokia.maps.geo.Coordinate} This property holds the object containing the geographic coordinates of the given info bubble.
node: {DomElement} This property holds the reference to the DOM node that of the bubble itself and all its contents.
xAlignment: {String} This property defines the actual current horizontal alignment of the info bubble.
yAlignment: {String} This property defines the actual current vertical alignment of the info bubble.

Table 197: Method Summary

Methods
close () This method closes a bubble and removes it from the map.
getState () : {String}

here

Methods

This method returns current state of the bubble:

- "initialized" for the bubble that is initialized (by [nokia.maps.map.component.InfoBubbles#initBubble](#)), but not opened yet
- "opened" for the bubble that is opened (by [nokia.maps.map.component.InfoBubbles#openBubble](#) or [nokia.maps.map.component.InfoBubbles.Bubble#open](#))
- "closed" for the bubble that is closed (by [nokia.maps.map.component.InfoBubbles#closeBubble](#) or [nokia.maps.map.component.InfoBubbles.Bubble#close](#))

isVisible () : {Boolean}

This method returns true if the bubble is fully or partially visible on the screen, and false if the bubble is fully hidden.

open (coordinate)

This method opens a previously initialized bubble on the map.

update (content, coordinate)

This method updates a bubble with new content and/or coordinates.

Interface Description

This interface defines a bubble which is shown on the map by an instance of [nokia.maps.map.component.InfoBubbles](#).

Property Details

contentNode: {DomElement}

This property holds the reference to the DOM node with the content of the bubble.

coordinate: {[nokia.maps.geo.Coordinate](#)}

This property holds the object containing the geographic coordinates of the given info bubble.

node: {DomElement}

This property holds the reference to the DOM node that of the bubble itself and all its contents. It is possible to check this node `offsetWidth` or `offsetHeight` properties to get size of bubble.

xAlignment: {String}

This property defines the actual current horizontal alignment of the info bubble.

here

yAlignment: {String}

This property defines the actual current vertical alignment of the info bubble.

Method Details

close()

This method closes a bubble and removes it from the map.

getState(): {String}

This method returns current state of the bubble:

- "initialized" for the bubble that is initialized (by [nokia.maps.map.component.InfoBubbles#initBubble](#)), but not opened yet
- "opened" for the bubble that is opened (by [nokia.maps.map.component.InfoBubbles#openBubble](#) or [nokia.maps.map.component.InfoBubbles.Bubble#open](#))
- "closed" for the bubble that is closed (by [nokia.maps.map.component.InfoBubbles#closeBubble](#) or [nokia.maps.map.component.InfoBubbles.Bubble#close](#))

Returns:

{String}	One of the three possible states of the bubble: "initialized", "opened" or "closed"

isVisible(): {Boolean}

This method returns true if the bubble is fully or partially visible on the screen, and false if the bubble is fully hidden.

Returns:

{Boolean}	A flag indicating whether the bubble is visible or fully hidden.

open(coordinate)

This method opens a previously initialized bubble on the map.

Parameters:

coordinate:	{ nokia.maps.geo.Coordinate } [optional]

here

An object containing the geographic coordinates of the location, where the bubble's anchor is to be placed. It is not necessary specify coordinate if co-ordinatate was set by update method.

update(content, coordinate)

This method updates a bubble with new content and/or coordinates. Both of parameters are optional, call without them can be used for forced update of bubble's size and alignment.

Parameters:

content: {String | DomElement} [optional]

The content to be shown in the info bubble provided as an HTML string; note that flash content in the bubble can lead to the effect that the flash content overlaps other elements in the document

coordinate: {[nokia.maps.geo.Coordinate](#)} [optional]

An object containing the geographic coordinates of the location, where the anchor is to be placed

Interface: Options

This interface is a member of [nokia.maps.map.component.InfoBubbles](#).

Extends: [nokia.maps.util.OObject](#)

Interface Summary

This interface defines the properties (options or keys) for [nokia.maps.map.component.InfoBubbles#options](#) property.

[For full details, see [nokia.maps.map.component.InfoBubbles.Options](#)]

Table 198: Property Summary

Properties
autoClose: {Boolean} This property specifies whether the info bubbles component allows to have only one or multiple bubbles open at the same time. backgroundColor: {String} This property specifies the background color of info bubbles.

here

Properties

color: {String}

This property specifies the foreground color (text color) of info bubbles.

defaultHeight: {String | Number}

This property holds the preferred height of bubbles.

defaultWidth: {String | Number}

This property holds the preferred width of bubbles.

defaultXAlignment: {String}

This property holds the preferred horizontal alignment of bubbles.

defaultYAlignment: {String}

This property holds the preferred vertical alignment of bubbles.

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Interface Description

This interface defines the properties (options or keys) for [nokia.maps.map.component.InfoBubbles#options](#) property.

Property Details

autoClose: {Boolean}

This property specifies whether the info bubbles component allows to have only one or multiple bubbles open at the same time.

When this property is set to true the component will automatically close all bubbles when a new bubble is opened. When this property is set to false bubbles will not be closed.

Changes to this property will take effect only when a new info bubble is opened.

Default Value: true

backgroundColor: {String}

here

This property specifies the background color of info bubbles. Any bubble that is opened after setting this property will have the specified background color. The property accepts colors as strings in hexadecimal notation or `rgb(a)` notation.

Note: Setting this property has an effect only after the component was added to a display's component collection.

color: {String}

This property specifies the foreground color (text color) of info bubbles. Any bubble that is opened after setting this property will have the specified foreground color. The property accepts colors as strings in hexadecimal notation or `rgb(a)` notation.

Note: Setting this property has an effect only after the component was added to a display's component collection.

defaultHeight: {String | Number}

This property holds the preferred height of bubbles. The property can either have a numeric value defining the height of bubbles in pixels or its value can be the string constant "auto" ([nokia.maps.map.component.InfoBubbles#DIMENSION_AUTO](#)) to allow the bubble to manage its size on its own. The default value is "auto".

Should very large amounts of text are displayed in an info bubble, it is best to set the preferred width to a fixed size to avoid rendering artifacts.

Platform-dependent styling can also be applied to the value of this property as a string, but the result of this may not yield expected results.

defaultWidth: {String | Number}

This property holds the preferred width of bubbles. The property can either have a numeric value defining the width of bubbles in pixels or it can be the string constant "auto" ([nokia.maps.map.component.InfoBubbles#DIMENSION_AUTO](#)) to allow the bubble to manage its size on its own. The default value is "auto".

Should very large amounts of text are displayed in an info bubble, it is best to set the preferred width to a fixed size to avoid rendering artifacts.

Platform-dependent styling can also be applied to the value of this property as a string, but the result of this may not yield expected results.

here

defaultXAlignment: {String}

This property holds the preferred horizontal alignment of bubbles. The property can have one of the following three values:

- left ([nokia.maps.map.component.InfoBubbles#ALIGNMENT_LEFT](#))
- right ([nokia.maps.map.component.InfoBubbles#ALIGNMENT_RIGHT](#))

The default value is right.

defaultYAlignment: {String}

This property holds the preferred vertical alignment of bubbles. The property can have one of the following three values:

- above ([nokia.maps.map.component.InfoBubbles#ALIGNMENT_ABOVE](#))
- below ([nokia.maps.map.component.InfoBubbles#ALIGNMENT_BELOW](#))

The default value is below.

Class: KeyControl

This class is a member of [nokia.maps.map.component](#).

Extends: [nokia.maps.map.component.Component](#)

Class Summary

This class adds keyboard control to the map.

[For full details, see [nokia.maps.map.component.KeyControl](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

[mapDisplay](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :

[attach](#), [destroy](#), [detach](#), [getId](#)

here

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This class adds keyboard control to the map. In 2D and 3D map mode the arrow keys pan the map. Additionally, tilt and heading can be modified in 3D mode by holding SHIFT key and using the arrow keys. In Panorama mode the arrow keys change the view angle. The + and - keys adjust the zoom in all modes.

Constructor Details

nokia.maps.map.component.KeyControl()

This method initializes a new instance of KeyControl.

Class: OverlaysSelector

This class is a member of [nokia.maps.map.component](#).

Extends: [nokia.maps.map.component.Component](#)

Class Summary

This component allows the user to select predefined overlays.

[For full details, see [nokia.maps.map.component.OverlaysSelector](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

[mapDisplay](#)

Table 199: Method Summary

Methods
showOverlayButtons (btnNames, show) This method shows or hides buttons for specific overlays.

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :

here

attach, destroy, detach, getId

Inherited from class *nokia.maps.util.OObject* :

addObserver, get, remove, removeObserver, set

Class Description

OverlaysSelector is a component that supports activation and deactivation of map overlay buttons. These buttons allow the user to switch additional overlay layers on and off.

Constructor Details

nokia.maps.map.component.OverlaysSelector(overlays, props)

This method initializes an instance of OverlaysSelector.

Parameters:

overlays:	{String[]} [optional]
	An array of overlay buttons that are to be shown by this component; it determines which buttons appear on the map and thus which layers can be (de)activated by the user; the possible values are: "trafficflow", "trafficincidents" (not available on all platforms); if this optional parameter is omitted, all available buttons are shown
props:	{Object} [optional]
	An object which can contain properties and/or methods to be appended to instances of this class; it can be used to extend a component

Method Details

showOverlayButtons(btnNames, show)

This method shows or hides buttons for specific overlays. Note that overriding methods have to call this super method as follows:

Parameters:

btnNames:	{String[]} [optional]
	An array of strings containing the names of all the buttons to be shown or or hidden

here

show: {Boolean} [optional]

A value indicating if the named buttons are to be shown (true - default) or hidden (false)

Example:

```
nokia.maps.map.component.OverlaysSelector.prototype.showOverlayButtons.apply(this, arguments);
```

Interface: Properties

This interface is a member of [nokia.maps.map.component.OverlaysSelector](#).

Interface Summary

This interface defines the properties (keys) that can be passed to the [nokia.maps.map.component.OverlaysSelector](#) constructor.

[For full details, see [nokia.maps.map.component.OverlaysSelector.Properties](#)]

Table 200: Property Summary

Properties
disableTooltips: {Boolean} Property disables tooltips that appear when traffic incident marker is clicked on the map.

Interface Description

This interface defines the properties (keys) that can be passed to the [nokia.maps.map.component.OverlaysSelector](#) constructor.

Property Details

disableTooltips: {Boolean}

Property disables tooltips that appear when traffic incident marker is clicked on the map.

Class: Overview

This class is a member of [nokia.maps.map.component](#).

Extends: [nokia.maps.map.component.Component](#)

here

Class Summary

This component shows small overview map.

[For full details, see [nokia.maps.map.component.Overview](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

[mapDisplay](#)

Table 201: Method Summary

Methods
getMinimapDimensions () : {Object}
This method retrieves the defined dimensions (width and height) of the map overview panel.
showMap (node)
This method shows a mini map display instance in the DOM node provided by the caller.

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :

[attach](#), [destroy](#), [detach](#), [getId](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Table 202: Event Summary

Events
resize
The identifier for an event that is fired for each step of the resize animation.
resizeend
The identifier for an event that is fired when the mini map resizing animation ends.
resizestart
The identifier for an event that is fired when the mini map resizing animation starts.

here

Class Description

This class defines a panel showing an overview of the map visible viewport.

Constructor Details

nokia.maps.map.component.Overview(props)

This method creates a panel showing an overview of the map.

Parameters:

props: {Object} [optional]

An object that can contain properties / methods to be appended to instances of this class; the argument can be used to extend a component.

Method Details

getMinimapDimensions(): {Object}

This method retrieves the defined dimensions (width and height) of the map overview panel. NOTE: Override this method to define the width / height of the mini map.

Returns:

{Object} The dimensions of the map as an object with the following properties:

- width - the width of the overview panel in pixels
- height - the height of the overview panel in pixels

showMap(node)

This method shows a mini map display instance in the DOM node provided by the caller.

Parameters:

node: {DomElement}

A reference to the DOM node in which to show the mini map

Event Details

resize

here

The identifier for an event that is fired for each step of the resize animation.

Event Handler Parameters:

`nokia.maps.dom.Event`

resizeend

The identifier for an event that is fired when the mini map resizing animation ends.

Event Handler Parameters:

`nokia.maps.dom.Event`

resizestart

The identifier for an event that is fired when the mini map resizing animation starts.

Event Handler Parameters:

`nokia.maps.dom.Event`

Class: PublicTransport

This class is a member of [*nokia.maps.map.component*](#).

Extends: [*nokia.maps.map.component.Component*](#)

Class Summary

The class `PublicTransport` defines a component that allows the user to turn on/off the public transportation overlay.

[For full details, see [*nokia.maps.map.component.PublicTransport*](#)]

Property Summary

Directly Inherited Properties

Inherited from class [*nokia.maps.map.component.Component*](#) :

here

mapDisplay

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :

[attach](#), [destroy](#), [detach](#), [getId](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

The class `PublicTransport` defines a component that allows the user to turn on/off the public transportation overlay.

Constructor Details

`nokia.maps.map.component.PublicTransport()`

Initializes a new instance of `PublicTransport`

Class: *ScaleBar*

This class is a member of [nokia.maps.map.component](#).

Extends: [nokia.maps.map.component.Component](#)

Class Summary

This components shows a scale bar.

[For full details, see [nokia.maps.map.component.ScaleBar](#)]

Table 203: Property Summary

Properties
<code>showImperialUnits</code>: {Boolean} This property indicates whether imperial rather than metric units of measurements are to be used.

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

here

mapDisplay

Table 204: Method Summary

Methods
<i>calculateMPP</i> () : {Number} This method calculates a value representing how many meters correspond to each pixel in the map view.
<i>switchUnit</i> () This method toggles the measurement units, in which the distance is displayed.
<i>updateScale</i> () This method is called, when the scale of the map view was changed.

Directly Inherited Methods

Inherited from class *nokia.maps.map.component.Component* :

attach, *destroy*, *detach*, *getId*

Inherited from class *nokia.maps.util.OObject* :

addObserver, *get*, *remove*, *removeObserver*, *set*

Class Description

This component defines a panel that holds the scale bar. The scale bar is a ruler showing map distances relative to the current zoom level of the map. The measurement type can be changed from metric to imperial, which will be reflected in the distance measurements.

Constructor Details

nokia.maps.map.component.ScaleBar ()

This method initializes a new instance of *ScaleBar*

Property Details

showImperialUnits: {Boolean}

This property indicates whether imperial rather than metric units of measurements are to be used. *true* indicates imperial units, *false* indicates metric units.

Method Details

calculateMPP () : {Number}

here

This method calculates a value representing how many meters correspond to each pixel in the map view.

Returns:

{Number}	A value indicating how many meters correspond to each pixel in the map view.

switchUnit()

This method toggles the measurement units, in which the distance is displayed.

updateScale()

This method is called, when the scale of the map view was changed. It updates the map component to reflect these changes.

Class: Traffic

This class is a member of [nokia.maps.map.component](#).

Extends: [nokia.maps.map.component.Component](#)

Class Summary

The class `Traffic` defines a component that allows the user to turn on/off the traffic overlay.

[For full details, see [nokia.maps.map.component.Traffic](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

[mapDisplay](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :

[attach](#), [destroy](#), [detach](#), [getId](#)

here

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

The class `Traffic` defines a component that allows the user to turn on/off the traffic overlay.

Constructor Details

`nokia.maps.map.component.Traffic()`

This method initializes a new instance of `Traffic`.

Class: *TrafficIncidents*

This class is a member of [nokia.maps.map.component](#).

Extends: [nokia.maps.map.component.Component](#)

Class Summary

The class `TrafficIncidents` defines a component that allows the user to turn on/off the traffic incidents overlay.

[For full details, see [nokia.maps.map.component.TrafficIncidents](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

[mapDisplay](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :

[attach](#), [destroy](#), [detach](#), [getId](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

here

Class Description

The class `TrafficIncidents` defines a component that allows the user to turn on/off the traffic incidents overlay.

Constructor Details

`nokia.maps.map.component.TrafficIncidents()`

This method initializes a new instance of `TrafficIncidents`.

Interface: `Properties`

This interface is a member of [nokia.maps.map.component.TrafficIncidents](#).

Interface Summary

This interface defines the properties (keys) that can be passed to the [nokia.maps.map.component.TrafficIncidents](#) constructor.

[For full details, see [nokia.maps.map.component.TrafficIncidents.Properties](#)]

Table 205: Property Summary

Properties
<code>disableButton</code> : {Boolean} Property disables traffic incidents button.
<code>disableTooltips</code> : {Boolean} Property disables tooltips that appear when traffic incident marker is clicked on the map.

Interface Description

This interface defines the properties (keys) that can be passed to the [nokia.maps.map.component.TrafficIncidents](#) constructor.

Property Details

`disableButton`: {Boolean}

Property disables traffic incidents button.

`disableTooltips`: {Boolean}

Property disables tooltips that appear when traffic incident marker is clicked on the map.

here

Class: *TypeSelector*

This class is a member of [nokia.maps.map.component](#).

Extends: [nokia.maps.map.component.Component](#)

Class Summary

This component allows user to select one of base map types.

[For full details, see [nokia.maps.map.component.TypeSelector](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

[mapDisplay](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :

[attach](#), [destroy](#), [detach](#), [getId](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This class defines a user interface component that allows the user to change the current base map type. Note that the component might not show all of the available base map types. Please check [nokia.maps.map.Display#availableBaseMapTypes](#) to see the complete list of map types.

To instantiate this class, use the default constructor without arguments:

```
var myTypeSelector = new nokia.maps.map.component.TypeSelector();
```

Constructor Details

`nokia.maps.map.component.TypeSelector()`

This method initializes a new instance of `TypeSelector`.

here

Class: ViewControl

This class is a member of [nokia.maps.map.component](#).

Extends: [nokia.maps.map.component.Component](#)

Class Summary

This class enables user to manipulate map, like changing heading tilt and center of the map.

[For full details, see [nokia.maps.map.component.ViewControl](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

[mapDisplay](#)

Table 206: Method Summary

Methods
updateHeading () This method is called when the map heading (bearing) changes.
updateTilting () This method is called when the map tilt changes.

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :

[attach](#), [destroy](#), [detach](#), [getId](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

The class ViewControl defines a panel that provides controls for heading (bearing), tilt and pan (setting the center position). The heading/tilt/pan component controls the displayed map orientation, i.e. North, South, East, West, and the angle map tilt.

here

Note that the heading/tilt/pan panel is only available in the Web version of the API. In device scenarios, adding this component has no effect.

Constructor Details

nokia.maps.map.component.ViewControl(props)

This method initializes an instance of `ViewControl` using the properties supplied by the caller.

Parameters:

props: `{Object}` [optional]
an object which can contain properties / methods to be appended to instances of this class; the object extends the component.

Method Details

updateHeading()

This method is called when the map heading (bearing) changes. It updates the map component to reflect these changes.

updateTilting()

This method is called when the map tilt changes. It updates the the map component to reflect these changes.

Class: ZoomBar

This class is a member of [nokia.maps.map.component](#).

Extends: [nokia.maps.map.component.Component](#)

Class Summary

This component enables zoom in and out buttons.

[For full details, see [nokia.maps.map.component.ZoomBar](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

here

mapDisplay

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :

[attach](#), [destroy](#), [detach](#), [getId](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

The class ZoomBar defines a component that allows the user to control the zoom level of the map. In the basic version, the zoom bar consists of two buttons (labeled "+" and "-") that permit the map user to change the zoom level.

To instantiate this class, use the default constructor without arguments:

```
var myZoomBar = new nokia.maps.map.component.ZoomBar();
```

Constructor Details

nokia.maps.map.component.ZoomBar()

This method initializes a new instance of ZoomBar.

Class: ZoomRectangle

This class is a member of [nokia.maps.map.component](#).

Extends: [nokia.maps.map.component.Component](#)

Class Summary

This component adds zoom rectangle functionality to the map.

[For full details, see [nokia.maps.map.component.ZoomRectangle](#)]

Table 207: Property Summary

Properties
on : {Boolean}

here

Properties

This property holds a value indicating if the zoom rectangle mode is currently activated (`true`) or not (`false`).

`rectNode`: {DomElement}

This property holds a reference to rectangle DOM element used for the zoom rectangle selection.

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

[mapDisplay](#)

Table 208: Method Summary

Methods

static **`switchOnOff`** ()

This method toggles the zoom rectangle functionality.

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :

[attach](#), [destroy](#), [detach](#), [getId](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This map component adds zoom rectangle functionality. When the component is activated, the user can zoom into an area by dragging a rectangle on the map.

Constructor Details

`nokia.maps.map.component.ZoomRectangle`(props)

This method initializes a new instance of `ZoomRectangle`.

Parameters:

props: {Object} [optional]
 optional initialization parameters

here

Property Details

on: {Boolean}

This property holds a value indicating if the zoom rectangle mode is currently activated (`true`) or not (`false`).

rectNode: {DomElement}

This property holds a reference to rectangle DOM element used for the zoom rectangle selection. By default, the element is a `<div>` with a border: "2px solid black", unless otherwise specified.

Method Details

static **switchOnOff()**

This method toggles the zoom rectangle functionality.

Interface: Properties

This interface is a member of [`nokia.maps.map.component.ZoomRectangle`](#).

Interface Summary

This interface defines the properties (keys) that can be passed to the [`nokia.maps.map.component.ZoomRectangle`](#) constructor.

[For full details, see [`nokia.maps.map.component.ZoomRectangle.Properties`](#)]

Interface Description

This interface defines the properties (keys) that can be passed to the [`nokia.maps.map.component.ZoomRectangle`](#) constructor.

Namespace: objects

This namespace is a member of [`nokia.maps.map.component`](#).

Namespace Summary

This namespace implements draggable behavior for map markers.

Namespace Description

This namespace implements draggable behavior for map markers.

here

Class: DragMarker

This class is a member of [nokia.maps.map.component.objects](#).

Extends: [nokia.maps.map.component.Component](#)

Class Summary

This component allows markers to be draggable.

[For full details, see [nokia.maps.map.component.objects.DragMarker](#)]

Table 209: Property Summary

Properties
enabled: {Boolean} This Boolean property is set to <code>true</code> if the component is enabled, <code>false</code> otherwise.

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

[mapDisplay](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :

[attach](#), [destroy](#), [detach](#), [getId](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This component implements the default draggability for markers. If this component is added to the map, all markers and standard markers on which the property "isDraggable" is set to `true` can be dragged by the user.

Constructor Details

[nokia.maps.map.component.objects.DragMarker](#)

here

This method initializes an instance of `DragMarker`, using the properties supplied by the caller. The method is a virtual constructor for the component.

Parameters:

`props`: {Object}

The default observable key-value pairs for this component (see `nokia.maps.util.OObject`).

Property Details

enabled: {Boolean}

This Boolean property is set to `true` if the component is enabled, `false` otherwise.

Namespace: *panning*

This namespace is a member of [*nokia.maps.map.component*](#).

Namespace Summary

This namespace implements panning functionality, including kinetic panning.

Namespace Description

This namespace implements panning functionality, including kinetic panning. It makes panning possible when the user clicks on the map or drags it.

Class: `Click`

This class is a member of [*nokia.maps.map.component.panning*](#).

Extends: [*nokia.maps.map.component.Component*](#)

Class Summary

This component pans the map when the user clicks in the map.

[For full details, see [*nokia.maps.map.component.panning.Click*](#)]

Table 210: Property Summary

Properties
enabled: {Boolean} This property determines if <code>Click</code> is enabled (<code>true</code>) or not (<code>false</code>).

here

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

[mapDisplay](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :

[attach](#), [destroy](#), [detach](#), [getId](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This component pans the map when the user clicks in the map.

Constructor Details

nokia.maps.map.component.panning.Click(props)

This method initializes an instance of `Click`, using the properties supplied by the caller. The method is a virtual constructor for the component.

Parameters:

props: {Object}

The default observable key-value pairs for this component (see [nokia.maps.util.OObject](#)).

Property Details

enabled: {Boolean}

This property determines if `Click` is enabled (true) or not (false).

Class: Drag

This class is a member of [nokia.maps.map.component.panning](#).

Extends: [nokia.maps.map.component.Component](#)

here

Class Summary

This component enables map dragging with the left mouse button or with the finger on a touch screen.

[For full details, see [nokia.maps.map.component.panning.Drag](#)]

Table 211: Property Summary

Properties
enabled: {Boolean} This property indicates if the drag component is enabled.
useKineticPanning: {Boolean} This property indicates whether the drag component uses kinetic panning if it is available.

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :
[mapDisplay](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :
[attach](#), [destroy](#), [detach](#), [getId](#)

Inherited from class [nokia.maps.util.OObject](#) :
[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This component enables map dragging with the left mouse button or with the finger on a touch screen.

Constructor Details

`nokia.maps.map.component.panning.Drag(props)`

This method initializes an instance of Drag, using the properties supplied by the caller. The method is a virtual constructor for the component.

Parameters:

here

props: {Object}

The default observable key-value pairs for this component (see `nokia.maps.util.OObject`).

Property Details

enabled: {Boolean}

This property indicates if the drag component is enabled. The value is `true` if the component is enabled, `false` otherwise.

useKineticPanning: {Boolean}

This property indicates whether the drag component uses kinetic panning if it is available. The value is `true` if kinetic panning is used, `false` otherwise.

Class: `Kinetic`

This class is a member of [*`nokia.maps.map.component.panning`*](#).

Extends: [*`nokia.maps.map.component.Component`*](#)

Class Summary

This component enables kinetic panning functionality on the map.

[For full details, see [*`nokia.maps.map.component.panning.Kinetic`*](#)]

Property Summary

Directly Inherited Properties

Inherited from class [*`nokia.maps.map.component.Component`*](#) :

[*`mapDisplay`*](#)

Method Summary

Directly Inherited Methods

Inherited from class [*`nokia.maps.map.component.Component`*](#) :

[*`attach`*](#), [*`destroy`*](#), [*`detach`*](#), [*`getId`*](#)

Inherited from class [*`nokia.maps.util.OObject`*](#) :

here

addObserver, get, remove, removeObserver, set

Class Description

This component adds kinetic panning functionality to the map display. If it is added to a display, other components in the panning namespace are able automatically to detect it and use kinetic panning. These components include `panning.Click`, `panning.Tap` and `panning.Drag`.

Kinetic panning consists of two phases: move and slide. During the first phase, map movements and their timing are recorded so that the kinetic panning module can determine how fast (pixels per second) and in which direction the map moves. The sliding phase begins when the moving map is released. The map then continues to move in the same direction at a gradually diminishing speed (as if friction were slowing it down), until it comes to a halt.

Kinetic panning from other components can be used as an alternative, but not at the same time as this component: `nokia.maps.map.component.panning.Kinetic` implements mechanisms to ensure that only one kinetic panning is performed at a time. The example below demonstrates how a kinetic panning is implemented:

```
// Query for the kinetic component.
var kineticComponent = display.getComponentById("panning.Kinetic");

// Initialize the kinetic engine to record a new movement -- this
// starts the movement phase.
var kineticMovement = kineticComponent.startMovement();

// Move the map. If the last argument is omitted, no movement occurs
// (simulation mode); the kinetic panning component simply records it
// for the sliding phase calculations.
kineticMovement.move(47, 11, 08, 15, true, evt.timeStamp);
//...

// Finish the recording and release the map, which causes the
// sliding phase to start. You can't reuse a kinetic movement, so
// after calling endMovement(), all further calls to move() are
// ignored.
kineticMovement.endMovement();

// From now on the map continues to slide until another kinetic movement
// occurs
// or the map speed decelerates to zero due to simulated friction.
kineticMovement.addObserver(
    "finished",
    function(kineticMovement, key, value, oldValue) {
        if (value===true) {
            alert("kinetic movement "+
                (this.aborted?"was aborted":"finished successfully")+
                "!");
        }
    },
    {});
```

here

```
this
);
```

Constructor Details

nokia.maps.map.component.panning.Kinetic(props)

This method initializes an instance of **Kinetic**, using the properties supplied by the caller. The method is a virtual constructor for the component.

Parameters:

props: {Object}

The default observable key-value pairs for this component (see [nokia.maps.util.OObject](#)).

Class: Tap

This class is a member of [nokia.maps.map.component.panning](#).

Extends: [nokia.maps.map.component.Component](#)

Class Summary

This component pans the map when the user taps in the map.

[For full details, see [nokia.maps.map.component.panning.Tap](#)]

Table 212: Property Summary

Properties
enabled: {Boolean} This property determines if Tap is enabled (true) or not (false).
useKineticPanning: {Boolean} This property determines if Tap is to use kinetic panning when available (true) or not (false).

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

[mapDisplay](#)

Method Summary

Directly Inherited Methods

here

Inherited from class [nokia.maps.map.component.Component](#) :

[attach](#), [destroy](#), [detach](#), [getId](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This component pans the map when the user taps in the map.

Constructor Details

nokia.maps.map.component.panning.Tap(props)

This method initializes an instance of Tap, using the properties supplied by the caller. The method is a virtual constructor for the component.

Parameters:

props: {Object}

The default observable key-value pairs for this component (see [nokia.maps.util.OObject](#)).

Property Details

enabled: {Boolean}

This property determines if Tap is enabled (`true`) or not (`false`).

useKineticPanning: {Boolean}

This property determines if Tap is to use kinetic panning when available (`true`) or not (`false`).

Namespace: zoom

This namespace is a member of [nokia.maps.map.component](#).

Namespace Summary

This namespace implements zooming functionality, allowing the map zoom to change when the user clicks or taps on the map, scrolls the mouse wheel or uses gestures.

here

Namespace Description

This namespace implements zooming functionality, allowing the map zoom to change when the user clicks or taps on the map, scrolls the mouse wheel or uses gestures.

Class: `DoubleClick`

This class is a member of [`nokia.maps.map.component.zoom`](#).

Extends: [`nokia.maps.map.component.Component`](#)

Class Summary

This class represents a component that zooms the map when the user double clicks into it.

[For full details, see [`nokia.maps.map.component.zoom.DoubleClick`](#)]

Table 213: Property Summary

Properties
<code>enabled</code>: {Boolean} This property determines if the component is enabled (<code>true</code>) or disabled (<code>false</code>).

Directly Inherited Properties

Inherited from class [`nokia.maps.map.component.Component`](#) :

[`mapDisplay`](#)

Method Summary

Directly Inherited Methods

Inherited from class [`nokia.maps.map.component.Component`](#) :

[`attach`](#), [`destroy`](#), [`detach`](#), [`getId`](#)

Inherited from class [`nokia.maps.util.OObject`](#) :

[`addObserver`](#), [`get`](#), [`remove`](#), [`removeObserver`](#), [`set`](#)

Class Description

This class represents a component that zooms the map when the user double clicks into it. It zooms in on a left mouse button double click and zooms out on a right mouse button double click.

here

Constructor Details

`nokia.maps.map.component.zoom.DoubleClick(props)`

This method initializes an instance of `DoubleClick`, using the properties supplied by the caller. The method is a virtual constructor for the component.

Parameters:

`props`: {Object}
The default observable key-value pairs for this component (see `nokia.maps.util.OObject`).

Property Details

enabled: {Boolean}

This property determines if the component is enabled (`true`) or disabled (`false`). When `nokia.maps.map.component.zoom.DoubleClick` is disabled, zooming in response to mouse double clicks does not occur.

Class: `DoubleTap`

This class is a member of [`nokia.maps.map.component.zoom`](#).

Extends: [`nokia.maps.map.component.Component`](#)

Class Summary

This class represents a component that zooms in the map view when the user taps twice on the map on a touch screen.

[For full details, see [`nokia.maps.map.component.zoom.DoubleTap`](#)]

Table 214: Property Summary

Properties
<code>enabled</code> : {Boolean} This property determines if the component is enabled (<code>true</code>) or disabled (<code>false</code>).

Directly Inherited Properties

Inherited from class [`nokia.maps.map.component.Component`](#) :

[`mapDisplay`](#)

here

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :

[attach](#), [destroy](#), [detach](#), [getId](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This class represents a component that zooms in the map view when the user taps twice on the map on a touch screen.

Constructor Details

nokia.maps.map.component.zoom.DoubleTap(props)

This method initializes an instance of DoubleTap, using the properties supplied by the caller. The method is a virtual constructor for the component.

Parameters:

props:	{Object}
	The default observable key-value pairs for this component (see nokia.maps.util.OObject).

Property Details

enabled: {Boolean}

This property determines if the component is enabled (true) or disabled (false). When [nokia.maps.map.component.zoom.DoubleTap](#) is disabled, zooming on double tap does not occur.

Class: Gesture

This class is a member of [nokia.maps.map.component.zoom](#).

Extends: [nokia.maps.map.component.Component](#)

Class Summary

This component will zoom the map if the user makes a zoom gesture at a touch screen.

here

[For full details, see [nokia.maps.map.component.zoom.Gesture](#)]

Table 215: Property Summary

Properties
enabled : {Boolean} If this property is false, then the component will be disabled and no zoom will occur.

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :
[mapDisplay](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :
[attach](#), [destroy](#), [detach](#), [getId](#)

Inherited from class [nokia.maps.util.OObject](#) :
[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This component will zoom the map if the user makes a zoom gesture at a touch screen.

Constructor Details

nokia.maps.map.component.zoom.Gesture()

This method initializes an instance of Gesture.

Property Details

enabled: {Boolean}

If this property is false, then the component will be disabled and no zoom will occur.

Class: MouseWheel

This class is a member of [nokia.maps.map.component.zoom](#).

Extends: [nokia.maps.map.component.Component](#)

here

Class Summary

This class represents a component that zooms in the map view when the user taps twice on the map on a touch screen.

[For full details, see [nokia.maps.map.component.zoom.MouseWheel](#)]

Table 216: Property Summary

Properties
enabled : {Boolean} This property determines if the component is enabled (true) or disabled (false).

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

[mapDisplay](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :

[attach](#), [destroy](#), [detach](#), [getId](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This class represents a component that zooms in the map view when the user taps twice on the map on a touch screen.

Constructor Details

nokia.maps.map.component.zoom.MouseWheel()

This method initializes an instance of MouseWheel.

Property Details

enabled: {Boolean}

here

This property determines if the component is enabled (`true`) or disabled (`false`). When `nokia.maps.map.component.zoom.MouseWheel` is disabled, zooming in response to double taps does not occur.

Namespace: provider

This namespace is a member of [nokia.maps.map](#).

Namespace Summary

This namespace provides basic provider classes that can be extended and used for providing own tiles or objects to map.

Namespace Description

This namespace provides basic provider classes that can be extended and used for providing own tiles or objects to map.

Class: BitmapProvider

This class is a member of [nokia.maps.map.provider](#).

Extends: [nokia.maps.map.provider.Provider](#)

Class Summary

This is an class representing a bitmap provider.

[For full details, see [nokia.maps.map.provider.BitmapProvider](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.provider.Provider](#) :

[description](#), [getInvalidationMark](#), [id](#), [label](#), [max](#), [min](#)

Table 217: Method Summary

Methods
create (id, bBox, cacheOnly) : {nokia.maps.map.provider.BitmapProvider.Request} This method creates bitmap data. getId () : {String}

here

Methods

This method returns created request ID.

request (bBox, cacheOnly, callback) : {*nokia.maps.map.provider.BitmapProvider.Request* | undefined}

Method requests an bitmap specified by getUrl method if given screen bounding box intersects with provider bounding box.

Directly Inherited Methods

Inherited from class *nokia.maps.map.provider.Provider* :

getCopyrights, providesLevel, shutdown, update

Inherited from class *nokia.maps.util.EventTarget* :

addListener, dispatch, removeListener

Event Summary

Directly Inherited Events

Inherited from class *nokia.maps.map.provider.Provider* :

response, update

Class Description

This provider is helpful to display bitmap in given bounding box. When updateCycle property is set than bitmap is periodically exchanged.

Constructor Details

nokia.maps.map.provider.BitmapProvider(options)

This method initializes the bitmap provider, using the caller-supplied options.

Parameters:

options: {*nokia.maps.map.provider.BitmapProvider.Options*}

Method Details

create(id, bBox, cacheOnly):

{*nokia.maps.map.provider.BitmapProvider.Request*}

This method creates bitmap data.

here

Parameters:

- id:** {String}
- Id of created bitmap request.
- bBox:** {*nokia.maps.geo.BoundingBox*}
- BoundingBox returned by
nokia.maps.map.provider.BitmapProvider#getBoundingBox method.
- cacheOnly:** {Boolean}
- A flag to signal that bitmap is created in cacheOnly mode, see
nokia.maps.map.provider.BitmapProvider#request method.

Returns:

{*nokia.maps.map.provider.BitmapProvider.Request*}

getId(): {String}

This method returns created request ID.

Returns:

{String}

request(bBox, cacheOnly, callback):

{*nokia.maps.map.provider.BitmapProvider.Request* | undefined}

Method requests an bitmap specified by getUrl method if given screen bounding box intersects with provider bounding box.

Parameters:

- bBox:** {*nokia.maps.geo.BoundingBox*}
- Screen bounding box.
- cacheOnly:** {boolean} [optional, default: false]
- A flag to signal that only cached bitmaps should be taken into account

here

callback: {function([nokia.maps.map.provider.BitmapProvider.Request](#))}

Callback given by layer called when bitmap request was finished.

Returns:

{[nokia.maps.map.provider.BitmapProvider.Request](#)
| undefined}

The Request object which is handling bitmap image creation or undefined to signal "non provided" for request which are not covered by provider bounding box.

Interface: Options

This interface is a member of [nokia.maps.map.provider.BitmapProvider](#).

Interface Summary

This interface defines properties of class `BitmapProvider`

[For full details, see [nokia.maps.map.provider.BitmapProvider.Options](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.provider.Provider.Options](#) :

[description](#), [getCopyrights](#), [label](#), [max](#), [min](#), [updateCycle](#)

Table 218: Method Summary

Methods
getBoundingBox () : { nokia.maps.geo.BoundingBox } Returns the bounding box of a provided image.

Interface Description

This interface defines properties of class `BitmapProvider`

Method Details

getBoundingBox(): {[nokia.maps.geo.BoundingBox](#)}

here

Returns the bounding box of a provided image.

Returns:

`{nokia.maps.geo.BoundingBox}`

Class: Request

This class is a member of *nokia.maps.map.provider.BitmapProvider*.

Class Summary

This is an class representing a bitmap provider request object.

[For full details, see *nokia.maps.map.provider.BitmapProvider.Request*]

Table 219: Property Summary

Properties
<i>bbbox</i> : { <i>nokia.maps.geo.BoundingBox</i> }
Holds bounding box for which request was triggered.
<i>data</i> : {*}
Holds the data of Request.
<i>id</i> : {String}
Holds id of request.
<i>state</i> : {Boolean undefined}
Holds the state of the request.

Table 220: Method Summary

Methods
<i>abort</i> ()
Aborts the ongoing bitmap image request
<i>send</i> (onStateChange)
Fires request to crate new bitmap image

Class Description

This is an helper class which holds all necessary information to finalize bitmap request (paint it or request from server).

here

Constructor Details

nokia.maps.map.provider.BitmapProvider.Request(id, bbox, provider)

This method initializes the bitmap provider request.

Parameters:

id:	{String}
	Request id
bbox:	{ <i>nokia.maps.geo.BoundingBox</i> }
	Requested bounding box
provider:	{ <i>nokia.maps.map.provider.Provider</i> }
	Provider which create instance of request

Property Details

bbox: {*nokia.maps.geo.BoundingBox*}

Holds bounding box for which request was triggered.

data: {***}

Holds the data of Request.

id: {String}

Holds id of request.

state: {Boolean | undefined}

Holds the state of the request. Value undefined means undefined state. Value true means that request was successful. Value false means that request wasn't successful.

Method Details

abort()

Aborts the ongoing bitmap image request

here

send(onStateChange)

Fires request to create new bitmap image

Parameters:

onStateChange: {Function([nokia.maps.map.provider.BitmapProvider.Request](#))}

Callback called when state changed

Class: *CanvasProvider*

This class is a member of [nokia.maps.map.provider](#).

Extends: [nokia.maps.map.provider.BitmapProvider](#)

Class Summary

This class represents a provider a canvas object that supports drawing.

[For full details, see [nokia.maps.map.provider.CanvasProvider](#)]

Table 221: Property Summary

Properties
<i>draw</i>: This method draws an image on the canvas specified by the caller. <i>getBoundingBox</i>: {nokia.maps.geo.BoundingBox} This function retrieves the bounding box of a provided image.

Directly Inherited Properties

Inherited from class [nokia.maps.map.provider.Provider](#) :

[description](#), [getInvalidationMark](#), [id](#), [label](#), [max](#), [min](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.provider.BitmapProvider](#) :

[create](#), [getId](#), [request](#)

Inherited from class [nokia.maps.map.provider.Provider](#) :

here

getCopyrights, providesLevel, shutdown, update

Inherited from class *nokia.maps.util.EventTarget* :

addListener, dispatch, removeListener

Event Summary

Directly Inherited Events

Inherited from class *nokia.maps.map.provider.Provider* :

response, update

Class Description

This class represents a provider a canvas object that supports drawing.

Constructor Details

nokia.maps.map.provider.CanvasProvider(options)

This method initializes the canvas provider using the caller-supplied options.

Parameters:

options: {*nokia.maps.map.provider.CanvasProvider.Options*}

An object containing instance initialization parameters.

Example:

```
map = new nokia.maps.map.Display(...);
var cycleIndex = 0,
    canvasProvider = new nokia.maps.map.provider.CanvasProvider({
        min: 0,
        max: 20,
        opacity: 0.5,
        updateCycle: 1/10, //seconds
        draw: function(canvas, callback) {
            var ctx = canvas.getContext("2d");
            canvas.width = 500;
            canvas.height = 500;

            ctx.fillStyle = "rgb(200,0,0)";
            ctx.fillRect (0, 0, canvas.width, canvas.height);

            ctx.font = "100pt Arial";
            ctx.fillStyle = "rgb(200,200,0)";
            ctx.fillText("" + (1+cycleIndex), 10, 110);
```

here

```
// The following call is mandatory, it triggers the
// actual render on the map
callback(canvas, true, true);
},
getBoundingBox: function () {
  return new nokia.maps.geo.BoundingBox([54.992047222,
1.912952778],
                                         [46.855536111,
15.898755556])
}
});

canvasProvider.addListener("update", function (evt) {
  cycleIndex = ++cycleIndex % 10 || 0;
});

map.overlays.add(canvasProvider);
```

Property Details

draw:

This method draws an image on the canvas specified by the caller. The method must be supplied by CanvasProvider instances.

getBoundingBox: {[nokia.maps.geo.BoundingBox](#)}

This function retrieves the bounding box of a provided image. This function must be supplied by CanvasProvider instances.

Interface: Options

This interface is a member of [nokia.maps.map.provider.CanvasProvider](#).

Extends: [nokia.maps.map.provider.BitmapProvider.Options](#)

Interface Summary

This interface defines properties of the class {@link nokia.maps.map.provider.CanvasProvider}.

[For full details, see [nokia.maps.map.provider.CanvasProvider.Options](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.provider.Provider.Options](#) :

[description](#), [getCopyrights](#), [label](#), [max](#), [min](#), [updateCycle](#)

here

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.provider.BitmapProvider.Options](#) :
[getBoundingBox](#)

Interface Description

This interface defines properties of the class {@link nokia.maps.map.provider.CanvasProvider}.

Class: Request

This class is a member of [nokia.maps.map.provider.CanvasProvider](#).

Extends: [nokia.maps.map.provider.BitmapProvider.Request](#)

Class Summary

This class represents a canvas provider request object.

[For full details, see [nokia.maps.map.provider.CanvasProvider.Request](#)]

Table 222: Property Summary

Properties
data : {HTMLCanvasElement} This property holds the data associated with the request, which is and instance of HTMLCanvasElement.

Directly Inherited Properties

Inherited from class [nokia.maps.map.provider.BitmapProvider.Request](#) :
[bbox](#), [data](#), [id](#), [state](#)

Table 223: Method Summary

Methods
abort () This method aborts a pending draw canvas request.
send (onStateChange) This method sends a request to crate new canvas image.

Directly Inherited Methods

here

Inherited from class *nokia.maps.map.provider.BitmapProvider.Request* :

abort, *send*

Class Description

This is a helper class which holds a reference to requested canvas, the related bounding box and a callback.

Constructor Details

nokia.maps.map.provider.CanvasProvider.Request(id, bbox, provider)

This method initializes the canvas provider request.

Parameters:

id:	{String}
	Request id
bbox:	{ <i>nokia.maps.geo.BoundingBox</i> }
	An object representing the requested bounding box
provider:	{ <i>nokia.maps.map.provider.Provider</i> }
	An object representing the provider to which the request is addressed.

Property Details

data: {HTMLCanvasElement}

This property holds the data associated with the request, which is an instance of HTMLCanvasElement.

Method Details

abort()

This method aborts a pending draw canvas request.

send(onStateChange)

This method sends a request to create new canvas image.

here

Parameters:

onStateChange: {Function([nokia.maps.map.provider.CanvasProvider.Request](#))}

A callback function to be invoked when the image state has changed.

Interface: *IData*

This interface is a member of [nokia.maps.map.provider](#).

Interface Summary

An interface for a data object which is provided by a [nokia.maps.map.Provider](#).

[For full details, see [nokia.maps.map.provider.IData](#)]

Table 224: Property Summary

Properties
<i>id</i> : {String} The ID of the provider data.

Table 225: Method Summary

Methods
<i>getInvalidations</i> (invalidationMark) : {Number} To obtain the invalidation states of a provider's data.

Interface Description

An interface for a data object which is provided by a [nokia.maps.map.Provider](#).

Property Details

id: {String}

The ID of the provider data. The ID starts with the ID of the provider which provided the data object followed by an underscore "_".

Method Details

getInvalidations(invalidationMark): {Number}

To obtain the invalidation states of a provider's data.

here

Parameters:

`invalidationMark`: {Number}

A `invalidationMark` to check against.

Returns:

{Number} a bit mask which specifies the invalid parts of the checked provider data. If no bit is set (value 0) the provider data is valid. The meaning of all bits has to be defined by concrete `Provider` implementation which provides the data.

Class: *ImageProvider*

This class is a member of [nokia.maps.map.provider](#).

Extends: [nokia.maps.map.provider.BitmapProvider](#)

Class Summary

This class represents an image provider.

[For full details, see [nokia.maps.map.provider.ImageProvider](#)]

Table 226: Property Summary

Properties
getBoundingBox: {nokia.maps.geo.BoundingBox} This method retrieves the bounding box of the image.
getUrl: {String} This method retrieves the URL of the image associated with the given instance <code>nokia.maps.map.provider.ImageProvider</code>.

Directly Inherited Properties

Inherited from class [nokia.maps.map.provider.Provider](#) :

[description](#), [getInvalidationMark](#), [id](#), [label](#), [max](#), [min](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.provider.BitmapProvider](#) :

here

create, getId, request

Inherited from class *nokia.maps.map.provider.Provider* :

getCopyrights, providesLevel, shutdown, update

Inherited from class *nokia.maps.util.EventTarget* :

addListener, dispatch, removeListener

Event Summary

Directly Inherited Events

Inherited from class *nokia.maps.map.provider.Provider* :

response, update

Class Description

This class represents an image provider. It displays an image in a bounding box. When the property `updateCycle` is set, then the image is periodically exchanged.

Constructor Details

nokia.maps.map.provider.ImageProvider(options)

This method initializes the image provider, using the caller-supplied options.

Parameters:

options: *{nokia.maps.map.provider.ImageProvider.Options}*

An object containing instance initialization parameters.

Example:

```
map = new nokia.maps.map.Display(...);
//create and ImageProvider
var imageIds = "image1.jpg image2.jpg image3.jpg image4.jpg".split(" "),
    numberOfImagesToCache = 4,
    cycleIndex = 0,
    rainRadar = new nokia.maps.map.provider.ImageProvider({
        min: 0,
        max: 20,
        getBoundingBox: function () {
            return new nokia.maps.geo.BoundingBox([54.992047222,
1.912952778],
                                                    [46.855536111,
15.898755556]);
        },
    },
```

here

```
getUrl: function () {  
    return "http://example.com/" + imageIds[cycleIndex];  
},  
updateCycle: 2/3, //seconds  
cache: new nokia.maps.util.Cache(numberOfImagesToCache)  
});  
  
rainRadar.addListener("update", function (evt) {  
    cycleIndex = ++cycleIndex % imageIds.length || 0;  
});  
  
map.overlays.add(rainRadar);
```

Property Details

getBoundingBox: {[nokia.maps.geo.BoundingBox](#)}

This method retrieves the bounding box of the image. This method must be supplied by instances of `nokia.maps.map.provider.ImageProvider`.

getUrl: {String}

This method retrieves the URL of the image associated with the given instance `nokia.maps.map.provider.ImageProvider`. This method must be supplied by instances of `nokia.maps.map.provider.ImageProvider`.

Interface: Options

This interface is a member of [nokia.maps.map.provider.ImageProvider](#).

Extends: [nokia.maps.map.provider.BitmapProvider.Options](#)

Interface Summary

This interface defines the properties of the class {@link `nokia.maps.map.provider.ImageProvider`}.

[For full details, see [nokia.maps.map.provider.ImageProvider.Options](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.provider.Provider.Options](#) :

[description](#), [getCopyrights](#), [label](#), [max](#), [min](#), [updateCycle](#)

here

Table 227: Method Summary

Methods
<i>draw</i> (callback) This method draws a new image on the specified canvas.
<i>getUrl</i> () : {String} This method retrieves the URL of the image associated with the given instance of ImageProvider.

Directly Inherited Methods

Inherited from class [nokia.maps.map.provider.BitmapProvider.Options](#) :

[getBoundingBox](#)

Interface Description

This interface defines the properties of the class {@link nokia.maps.map.provider.ImageProvider}.

Method Details

draw(callback)

This method draws a new image on the specified canvas. The method must be implemented by instances of {@link nokia.maps.map.provider.ImageProvider}.

Parameters:

callback: {Function(HTMLCanvasElement|Boolean|Boolean)}

Callback function to be called when the new image is drawn to canvas. Its parameters are:

- canvas (HTMLCanvasElement) - An object representing the canvas on which to draw
- drawn (Boolean) - true if drawing was successful, otherwise false. The value is set to false when nothing was drawn, for example, because no data were provided.
- synchronous (Boolean) - true if the callback is called synchronously, otherwise false

getUrl(): {String}

here

This method retrieves the URL of the image associated with the given instance of `ImageProvider`.

Returns:

`{String}` The URL of the image associated with the given instance of `ImageProvider`.

Class: `Request`

This class is a member of [nokia.maps.map.provider.ImageProvider](#).

Extends: [nokia.maps.map.provider.BitmapProvider.Request](#)

Class Summary

This is a class representing an image provider request object.

[For full details, see [nokia.maps.map.provider.ImageProvider.Request](#)]

Table 228: Property Summary

Properties
<i>data</i>: <code>{Image}</code> This property holds the data associated with the request, which is the image object.
<i>id</i>: <code>{String}</code> This property holds the id of the request.

Directly Inherited Properties

Inherited from class [nokia.maps.map.provider.BitmapProvider.Request](#) :

[bbox](#), [data](#), [id](#), [state](#)

Table 229: Method Summary

Methods
<i>abort</i> (<code>()</code>) This method aborts the pending image provider request.
<i>send</i> (<code>onStateChange</code>) This method sends a request for notification to the image provider.

Directly Inherited Methods

Inherited from class [nokia.maps.map.provider.BitmapProvider.Request](#) :

here

abort, send

Class Description

This is a helper class which holds a reference to the requested image and the related bounding box and callback.

Constructor Details

nokia.maps.map.provider.ImageProvider.Request(id, bbox, provider)

This method initializes an image provider request.

Parameters:

id:	{String}
	The request id for ImageTile.Request; the value is URL of the image.
bbox:	{ <i>nokia.maps.geo.BoundingBox</i> }
	An object representing the bounding box (containing the image).
provider:	{ <i>nokia.maps.map.provider.Provider</i> }
	An object representing the provider to which the request is addressed

Property Details

data: {Image}

This property holds the data associated with the request, which is the image object.

id: {String}

This property holds the id of the request. In this case the id is the URL of the image.

Method Details

abort()

This method aborts the pending image provider request.

send(onStateChange)

here

This method sends a request for notification to the image provider.

Parameters:

onStateChange: {Function([nokia.maps.map.provider.ImageProvider.Request](#))}

A callback function to be called when the image state has changed.

Class: [ImgTileProvider](#)

This class is a member of [nokia.maps.map.provider](#).

Extends: [nokia.maps.map.provider.TileProvider](#)

Class Summary

This class provides Image objects as map tiles.

[For full details, see [nokia.maps.map.provider.ImgTileProvider](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.provider.TileProvider](#) :

[pixelProjections](#)

Inherited from class [nokia.maps.map.provider.Provider](#) :

[description](#), [getInvalidationMark](#), [id](#), [label](#), [max](#), [min](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.provider.TileProvider](#) :

[cancel](#), [checkAddy](#), [create](#), [createId](#), [destroy](#), [request](#)

Inherited from class [nokia.maps.map.provider.Provider](#) :

[getCopyrights](#), [providesLevel](#), [shutdown](#), [update](#)

Inherited from class [nokia.maps.util.EventTarget](#) :

[addListener](#), [dispatch](#), [removeListener](#)

here

Event Summary

Directly Inherited Events

Inherited from class [nokia.maps.map.provider.Provider](#) :

[response](#), [update](#)

Class Description

A class to provide Image objects as tiles. These Implementation of a TileProvider tries to load Images from a URL which will be specified by the ImgTileProvider instance. The world is repeating on x-axis.

Constructor Details

nokia.maps.map.provider.ImgTileProvider(options)

Parameters:

options: {[nokia.maps.map.provider.ImgTileProvider.Options](#)}

Interface: Options

This interface is a member of [nokia.maps.map.provider.ImgTileProvider](#).

Interface Summary

The interface defines properties of class ImgTileProvider

[For full details, see [nokia.maps.map.provider.ImgTileProvider.Options](#)]

Table 230: Property Summary

Properties
timeout: {number} A timeout in milliseconds to define a duration within which a failed image load is treated as a valid server response.

Directly Inherited Properties

Inherited from class [nokia.maps.map.provider.TileProvider.Options](#) :

[alpha](#), [cache](#), [opacity](#), [scalable](#)

Inherited from class [nokia.maps.map.provider.Provider.Options](#) :

[description](#), [getCopyrights](#), [label](#), [max](#), [min](#), [updateCycle](#)

here

Table 231: Method Summary

Methods
<code>getUrl</code> (level, row, col) : {string} A method to create the URL for the specified tile.

Interface Description

The interface defines properties of class `ImgTileProvider`

Property Details

`timeout`: {number}

A timeout in milliseconds to define a duration within which a failed image load is treated as a valid server response. To never treat a failed image load as valid server response this value should be set to a negative value.

Default Value: 30000

Method Details

`getUrl`(level, row, col): {string}

A method to create the URL for the specified tile.

Parameters:

`level`: {number}

The zoom level of the requested tile URL.

`row`: {number}

The row of the requested tile URL.

`col`: {number}

The column of the requested tile URL.

Returns:

{string} The URL of the requested tile or `false` if the tile is not provided

here

Class: Provider

This class is a member of `nokia.maps.map.provider`.

Extends: `nokia.maps.util.EventTarget`

Class Summary

This is an abstract class that represents a map provider.

[For full details, see `nokia.maps.map.provider.Provider`]

Table 232: Property Summary

Properties
description : {String} This property holds a description of the provider to be used in the user interface, for example in tool tips.
getInvalidationMark : {Number} This method retrieves the last invalidation mark for the provider data.
readonly id : {Number} This property holds the id of the Provider instance.
label : {String} This property represents a label for the provider to be displayed in the user interface.
max : {Number} This property indicates the maximum zoom level.
min : {Number} This property indicates the minimum zoom level.

Table 233: Method Summary

Methods
getCopyrights (area, zoom) : {Object[]} This method retrieves the copyright information for a specified geographical area and zoom level.
providesLevel (zoomLevel) : {Boolean} This method checks if the zoom level supplied by the caller lies within the range supported by the given provider object.
shutdown () This method releases resources that were allocated by the provider.
update ()

here

Methods

This method increases the value of `invalidationMark` and publishes a message with topic "updated" asynchronously.

Directly Inherited Methods

Inherited from class [nokia.maps.util.EventTarget](#) :

[addListener](#), [dispatch](#), [removeListener](#)

Table 234: Event Summary

Events

[response](#)

This event is fired when an asynchronously processing of a request has been finished.

[update](#)

This event is fired if when the provider content has been changed.

Class Description

This is an abstract class representing a map provider. A provider can be understood as a data vendor for items such as:

- Map tiles - this is common functionality
- Spatial objects - for example to visualize traffic or public transport lines
- Marker objects - for example to mark stores in a store finder overlay

The constructor for this class initializes an object based on `Provider`, using the argument provided by the caller.

Constructor Details

`nokia.maps.map.provider.Provider(options)`

Parameters:

options: [{nokia.maps.map.provider.Provider.Options}](#)

Property Details

description: {String}

This property holds a description of the provider to be used in the user interface, for example in tool tips.

here

Default Value: ""

getInvalidationMark: {Number}

This method retrieves the last invalidation mark for the provider data.

See: [nokia.maps.map.provider.Provider#getInvalidations](#)

readonly **id:** {Number}

This property holds the id of the Provider instance.

label: {String}

This property represents a label for the provider to be displayed in the user interface.

Default Value: ""

max: {Number}

This property indicates the maximum zoom level. Its value must be an integer in range from -1 to 30. A value of -1 indicates that the provider doesn't provide data in any zoom level.

Default Value: -1

min: {Number}

This property indicates the minimum zoom level. Its value must be an integer in the range from 0 to 30.

Default Value: 0

Method Details

getCopyrights(area, zoom): {Object[]}

This method retrieves the copyright information for a specified geographical area and zoom level. This method can be defined by [nokia.maps.map.provider.Provider.Options#getCopyrights](#).

here

Parameters:

- area:** `{nokia.maps.geo.BoundingBox}`
The GEO area for which to obtain the copyright information.
- zoom:** `{Number}`
The zoom level for which to obtain the copyright information.

Returns:

- `{Object[]}` An array of objects, each containing the string elements "label" and "alt", for example [{ label: "DigitalGlobe 2009", alt: "copyright 2009 DigitalGlobe, Inc." }]

providesLevel(zoomLevel): {Boolean}

This method checks if the zoom level supplied by the caller lies within the range supported by the given provider object.

Parameters:

- zoomLevel:** `{Number}`
The zoom level to check.

Returns:

- `{Boolean}` true if the supplied zoom level is in range of [min ... max] zoom level supported by the provider, otherwise false.

shutdown()

This method releases resources that were allocated by the provider. Note that after the method has been called, the engine is unusable.

update()

This method increases the value of `invalidationMark` and publishes a message with topic "updated" asynchronously.

here

Event Details

response

This event is fired when an asynchronously processing of a request has been finished.

Event Handler Parameters:

evt {[nokia.maps.util.Event](#)}
An object representing the event.

update

This event is fired if when the provider content has been changed.

Event Handler Parameters:

evt {[nokia.maps.util.Event](#)}
An object representing the event

Interface: Options

This interface is a member of [nokia.maps.map.provider.Provider](#).

Interface Summary

This interface defines options for the class Provider constructor.

[For full details, see [nokia.maps.map.provider.Provider.Options](#)]

Table 235: Property Summary

Properties
description: {String} This property holds a description of the provider to be used in the user interface, for example in tool tips.
getCopyrights: {Function} This property is designed to hold a function that retrieves copyright information.
label: {String} This property represents a label for the provider, for example to indicate the type or overlay buttons.
max: {Number}

here

Properties

This property indicates the maximum zoom level.

min: {Number}

This property indicates the initial minimum zoom level.

updateCycle: {Number}

This property indicates the interval between automatic update cycles in seconds (integer).

Interface Description

This interface defines options for the class `Provider` constructor.

Property Details

description: {String}

This property holds a description of the provider to be used in the user interface, for example in tool tips.

getCopyrights: {Function}

This property is designed to hold a function that retrieves copyright information.

See: [nokia.maps.map.provider.Provider#getCopyrights](#)

label: {String}

This property represents a label for the provider, for example to indicate the type or overlay buttons.

max: {Number}

This property indicates the maximum zoom level. Its value must be an integer in range from -1 to 30. A value of -1 indicates that the provider doesn't provide data in any zoom level.

min: {Number}

This property indicates the initial minimum zoom level. Its value must be an integer in the range from 0 to 30.

here

updateCycle: {Number}

This property indicates the interval between automatic update cycles in seconds (integer). If the value is less than 1 the property is ignored. See also [nokia.maps.map.provider.Provider#update](#)

Class: TileProvider

This class is a member of [nokia.maps.map.provider](#).

Extends: [nokia.maps.map.provider.Provider](#)

Class Summary

This is an abstract base class representing a tile provider.

[For full details, see [nokia.maps.map.provider.TileProvider](#)]

Table 236: Property Summary

Properties
readonly pixelProjections : {Object}
This property holds a hash table of pixel projections for for all supported integral levels [min .

Directly Inherited Properties

Inherited from class [nokia.maps.map.provider.Provider](#) :

[description](#), [getInvalidationMark](#), [id](#), [label](#), [max](#), [min](#)

Table 237: Method Summary

Methods
cancel (id) : {nokia.maps.map.provider.Tile}
This method informs the TileProvider that a requested tile is no longer required.
checkAddy (level, row, col) : {Number}
This method checks if a tile address (level, row, column) is valid and, if necessary, adjusts the column (repeating world on x-axis).
create (level, row, col, id, cacheOnly) : {nokia.maps.map.provider.Tile}
This method creates a new tile.
createId (level, row, col, noCheck) : {String}
This method creates a unique tile id if the specified tile would provided.
destroy ()
This method destroys a TileProvider and frees resources.

here

Methods

request (level, row, col, cacheOnly) : {nokia.maps.map.provider.Tile | Boolean}

This method requests a specified tile.

Directly Inherited Methods

Inherited from class *nokia.maps.map.provider.Provider* :

getCopyrights, *providesLevel*, *shutdown*, *update*

Inherited from class *nokia.maps.util.EventTarget* :

addListener, *dispatch*, *removeListener*

Event Summary

Directly Inherited Events

Inherited from class *nokia.maps.map.provider.Provider* :

response, *update*

Class Description

This is an abstract base class representing a tile provider. Derived classes must implement the `create()` method.

Constructor Details

`nokia.maps.map.provider.TileProvider`(options)

This method initializes the tile provider, using the caller-supplied options.

Parameters:

options: {*nokia.maps.map.provider.TileProvider.Options*}

Property Details

readonly **`pixelProjections`**: {Object}

This property holds a hash table of pixel projections for for all supported integral levels [min ... max].

Method Details

`cancel`(id): {nokia.maps.map.provider.Tile}

here

This method informs the `TileProvider` that a requested tile is no longer required.

Parameters:

id: `{String}`
The id of the requested tile

Returns:

`{nokia.maps.map.provider.Tile}` A tile object whose loading has been cancelled, otherwise undefined.

checkAddy(level, row, col): `{Number}`

This method checks if a tile address (level, row, column) is valid and, if necessary, adjusts the column (repeating world on x-axis).

Parameters:

level: `{Number}`
The zoom level of the requested tile

row: `{Number}`
The row of the requested tile

col: `{Number}`
The column of the requested tile

Returns:

`{Number}` The adjusted column if the tile address is valid, otherwise -1

create(level, row, col, id, cacheOnly): `{nokia.maps.map.provider.Tile}`

This method creates a new tile. This method must be implemented by derived classes. The method is used internally by a `TileProvider` instance when a tile is requested.

Parameters:

level: `{Number}`

here

	The zoom level of the requested tile
row:	{Number}
	The row of the requested tile
col:	{Number}
	The column of the requested tile
id:	{String}
	The ID of the requested tile
cacheOnly:	{Boolean} [optional, default: false]
	A flag to signal that only cached tiles should take into account

Returns:

`{nokia.maps.map.provider.Tile}` The requested tile

createId(level, row, col, noCheck): {String}

This method creates a unique tile id if the specified tile would be provided. The id starts with the TileProvider id followed by an underscore ("_"). If the TileProvider uses the same ids for different tiles (for example repeating on x-axis), the id must be adjusted.

The standard implementation returns false for the following cases:

- if level is less than this.min
- if level is greater than this.max
- if row is negative
- if row is greater than 2^{level} (the numbers of tiles per axis for this level)

Otherwise it returns an ID in this format: "providerId_level_row_col", where col is adjusted to the range of [0 ... number_of_tiles_per_axis].

Parameters:

level:	{Number}
	The zoom level of the requested tile
row:	{Number}

here

	The row of the requested tile
<code>col:</code>	<code>{Number}</code>
	The column of the requested tile
<code>noCheck:</code>	<code>{Boolean}</code> [optional, default: false]
	A flag to signal that the ID should created without validation check
 Returns:	
<code>{String}</code>	The created ID or false if the requested tile is not provided

destroy()

This method destroys a `TileProvider` and frees resources.

request(level, row, col, cacheOnly): `{nokia.maps.map.provider.Tile | Boolean}`

This method requests a specified tile. IMPLEMENTATION HINTS:

- `cacheOnly` is `true`:
 - returns a `Tile`: `TileProvider` was able to create the requested `Tile` instantly.
 - returns `true`: The `Tile` ID found in `TileIdCache` but the `TileProvider` will provide the requested `Tile` delayed (see "response" event).
 - returns `false`: The `TileProvider` doesn't provide the requested `Tile` or the requested `Tile` was not found in `Cache`.
- `cacheOnly` was set to `false` (or omitted):
 - returns a `Tile`: `TileProvider` was able to create the requested `Tile` instantly.
 - returns `true`: The `TileProvider` will provide the requested `Tile` delayed.
 - returns `false` (or undefined): The `TileProvider` does not provide the requested `Tile`.
- The `TileProvider` should take into account that more than one customer could request one and the same tile. Therefore a separate counter for each item of the `requestedTiles` list is necessary. A `request()` call has to increase this counter.

Parameters:

<code>level:</code>	<code>{number}</code>
	The zoom level of the requested tile

here

row: {number}
The row of the requested tile

col: {number}
The column of the requested tile

cacheOnly: {boolean} [optional, default: false]
A flag to signal that only cached tiles should be taken into account

Returns:

`{nokia.maps.map.provider.TileProvider}` The requested tile if it could be created immediately or false to signal "not provided/not cached" or true to signal "will possibly provide with a delay".

See: `nokia.maps.map.provider.TileProvider#response`

Interface: Options

This interface is a member of [nokia.maps.map.provider.TileProvider](#).

Interface Summary

This interface defines properties of class `TileProvider`

[For full details, see [nokia.maps.map.provider.TileProvider.Options](#)]

Table 238: Property Summary

Properties
alpha: {Boolean} This property is a flag that indicates if the provided tiles use PNG alpha transparency.
cache: {nokia.maps.util.Cache} This property holds the tile cache.
opacity: {Number} This property holds the opacity as a value between 0 (transparent) and 1 (opaque) to use for all tiles of the given provider.
scalable: This property is a flag that indicates if the provided tiles may be scaled during zoom level changes of the map.

Directly Inherited Properties

Inherited from class [nokia.maps.map.provider.Provider.Options](#) :

here

description, getCopyrights, label, max, min, updateCycle

Interface Description

This interface defines properties of class `TileProvider`

Property Details

alpha: {Boolean}

This property is a flag that indicates if the provided tiles use PNG alpha transparency. This is needed to support PNG alpha transparency in older browsers.

Default Value: `false`

cache: {*nokia.maps.util.Cache*}

This property holds the tile cache. If omitted, a global three-level cache (optimized for tiles) is used. If the property is set to `null`, no cache is used.

Default Value: `nokia.maps.map.provider.TileProvider.prototype.cache`

opacity: {Number}

This property holds the opacity as a value between 0 (transparent) and 1 (opaque) to use for all tiles of the given provider.

Default Value: `1`

scalable:

This property is a flag that indicates if the provided tiles may be scaled during zoom level changes of the map.

Default Value: `true`

Namespace: positioning

This namespace is a member of *nokia.maps*.

here

Namespace Summary

This namespace defines facilities related to positioning and specifically an implementation of the W3C Geolocation API.

Namespace Description

This namespace defines facilities related to positioning and specifically an implementation of the W3C Geolocation API.

Class: Manager

This class is a member of [nokia.maps.positioning](#).

Extends: [nokia.maps.util.OObject](#)

Class Summary

This class provides a set of methods based on the W3C geolocation API specification.

[For full details, see [nokia.maps.positioning.Manager](#)]

Table 239: Method Summary

Methods
clearWatch (watchId) This method clears and stops the specified position watch.
getCurrentPosition (successCallback, errorCallback, options) This method retrieves the current position, if available, and fires an event containing the data.
watchPosition (successCallback, errorCallback, options) : {Number} This method starts periodic monitoring of the position sensor and fires events containing any new or changed data collected from the sensor.

Directly Inherited Methods

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This class provides a set of methods based on the specification of the W3C geolocation API. The API reference can be found [here](#).

here

Constructor Details

nokia.maps.positioning.Manager()

This method initializes an instance of `positioning.Manager`.

Method Details

clearWatch(watchId)

This method clears and stops the specified position watch.

Parameters:

watchId: {Number}
 represents the watcher ID returned from
[*nokia.maps.positioning.Manager#watchPosition*](#).

getCurrentPosition(successCallback, errorCallback, options)

This method retrieves the current position, if available, and fires an event containing the data.

Parameters:

successCallback: {Function}
 A function to call when the position data is available

errorCallback: {Function}
 A function to call when there is an error getting the position data

options: {Object}
 An anonymous object containing options for getting the position data

watchPosition(successCallback, errorCallback, options): {Number}

This method starts periodic monitoring of the position sensor and fires events containing any new or changed data collected from the sensor.

Parameters:

successCallback: {Function}

here

	A function to call when the position data is available
errorCallback:	{Function}
	A function to call when there is an error retrieving position data
options:	{Object}
	An anonymous object containing options for retrieving position data
Returns:	
{Number}	the watcher ID

Namespace: component

This namespace is a member of [nokia.maps.positioning](#).

Namespace Summary

This namespace provides facilities to obtain geo-positioning information.

Namespace Description

This namespace provides facilities to obtain geo-positioning information.

Class: Positioning

This class is a member of [nokia.maps.positioning.component](#).

Extends: [nokia.maps.map.component.Component](#)

Class Summary

This class represents a component which handles geolocation positioning from browser.

[For full details, see [nokia.maps.positioning.component.Positioning](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

[mapDisplay](#)

here

Table 240: Method Summary

Methods
<i>requestPosition</i> (successCallback, errorCallback) This method requests the current geo positioning information from the positioning manager.

Directly Inherited Methods

Inherited from class *nokia.maps.map.component.Component* :

attach, destroy, detach, getId

Inherited from class *nokia.maps.util.OObject* :

addObserver, get, remove, removeObserver, set

Class Description

This class represents a component that switches positioning on and off. It is currently only supported in the *nokia_generic* UI package. Due to the missing support of the *W3C Geolocation API* it is not working in all Internet Explorer versions up to version 8.

Constructor Details

nokia.maps.positioning.component.Positioning()

This method initializes a new instance of Positioning.

Method Details

requestPosition(successCallback, errorCallback)

This method requests the current geo positioning information from the positioning manager. When the position information has been retrieved, the map moves to the current position and an accuracy circle is drawn using the method *#showPosition()*.

Parameters:

successCallback: {Function}

If a position is found this function would be called.

errorCallback: {Function}

If error occurs this callback would be executed.

here

See: [nokia.maps.positioning.Manager](#)

Namespace: routing

This namespace is a member of [nokia.maps](#).

Namespace Summary

This namespace defines facilities for calculating routes and adding them to a map display.

Namespace Description

This namespace defines facilities for calculating routes and adding them to a map display.

Class: Manager

This class is a member of [nokia.maps.routing](#).

Extends: [nokia.maps.util.OObject](#)

Class Summary

This class acts as a proxy to the routing service and is responsible for storing the currently selected routes.

[For full details, see [nokia.maps.routing.Manager](#)]

Table 241: Property Summary

Properties
metricSystem: {String} This property holds the metric system information to be used by the backend when return maneuver instructions.
state: {String} This property holds the state of the request sent by the given instance of the Manager.

Table 242: Method Summary

Methods
calculateRoute (waypoints, mode) This method invokes the back-end calculateRoute service.
clear ()

here

Methods

This method clears all the stored routes.

destroy ()

This method cleans up the given instance of `routing.Manager` before releasing it from memory.

getErrorMessage () : {*nokia.maps.routing.ServiceError*}

This method returns the cause of an error.

getRoutes () : {*nokia.maps.routing.Route*[]}

This method retrieves routes that have been calculated by the back end.

Directly Inherited Methods

Inherited from class *nokia.maps.util.OObject* :

addObserver, *get*, *remove*, *removeObserver*, *set*

Class Description

This class acts as a proxy to the routing service and is responsible for storing the currently selected routes.

Constructor Details

nokia.maps.routing.Manager()

This method creates a new empty instance of the `Manager` class.

Property Details

metricSystem: {String}

This property holds the metric system information to be used by the backend when return maneuver instructions. It can be one of: undefined (default, metric system is determined by the backend), 'metric' or 'imperial'.

By default this value is undefined and can be set to one of the valid values before calculating a route in order to inform the routing backend to return routes with the specified metric system.

Example:

```
//force the routing backend to use distances in the metric system
routingManager.set('metricSystem', 'metric');
```

here

state: {String}

This property holds the state of the request sent by the given instance of the Manager. The possible values are: "initial", "started", "finished", "failed" and "cleared".

Method Details

calculateRoute(waypoints, mode)

This method invokes the back-end calculateRoute service.

Parameters:

waypoints: {*nokia.maps.routing.WaypointParameterList*}

A list of waypoints containing start, destination, and optional via points; the list represents the basis for a route calculation

mode: {*nokia.maps.routing.Mode*}

An object that defines the mode for the routes to be calculated

clear()

This method clears all the stored routes.

destroy()

This method cleans up the given instance of *routing.Manager* before releasing it from memory. It is important to call this method especially when you use asynchronous code, because it kills coroutines, thus preventing errors.

getErrorCause(): {*nokia.maps.routing.ServiceError*}

This method returns the cause of an error.

Returns:

{*nokia.maps.routing.ServiceError*}

The error object containing the error type

here

getRoutes(): {[nokia.maps.routing.Route](#)[]}}

This method retrieves routes that have been calculated by the back end.

Returns:

{[nokia.maps.routing.Route](#)[]}}

The routes returned by the back end.

Interface: Maneuver

This interface is a member of [nokia.maps.routing](#).

Interface Summary

This interface defines a maneuver, which describes an action that a person needs to perform when following a route.

[For full details, see [nokia.maps.routing.Maneuver](#)]

Table 243: Property Summary

Properties
direction: {String} This property holds a maneuver direction hint.
instruction: {String} This property contains a textual instruction describing the required maneuver, for example, "Turn left onto Minna St.
length: {Number} This property holds the length of the segment following the given maneuver until the next maneuver.
position: {nokia.maps.geo.Coordinate} This property holds an object containing the coordinates of the position, where the maneuver starts.
shape: {nokia.maps.geo.Shape} This property holds a shape object (a polyline) that represents the route segment following the given maneuver until the next maneuver.
travelTime: {Number} This property holds the time needed to travel the route segment following the given maneuver until the next maneuver.
type: {String} This property contains an identifier to differentiate public from private transport maneuvers.

here

Interface Description

This interface defines a maneuver, which describes an action that a person needs to perform when following a route. Specifically, a maneuver applies to a particular route link, which is a part of a route calculated by the back-end server, has a server-assigned identifier, and is defined as a set of instances of [nokia.maps.geo.Coordinate](#).

Property Details

direction: {String}

This property holds a maneuver direction hint. The possible values are:

- "forward"
- "bearRight"
- "lightRight"
- "right"
- "hardRight"
- "uTurnRight"
- "uTurnLeft"
- "hardLeft"
- "left"
- "lightLeft"
- "bearLeft"

instruction: {String}

This property contains a textual instruction describing the required maneuver, for example, "Turn left onto Minna St."

length: {Number}

This property holds the length of the segment following the given maneuver until the next maneuver.

position: {[nokia.maps.geo.Coordinate](#)}

This property holds an object containing the coordinates of the position, where the maneuver starts.

here

shape: {[nokia.maps.geo.Shape](#)}

This property holds a shape object (a polyline) that represents the route segment following the given maneuver until the next maneuver.

travelTime: {Number}

This property holds the time needed to travel the route segment following the given maneuver until the next maneuver. The routing type determines if the value of the property takes into account traffic information.

type: {String}

This property contains an identifier to differentiate public from private transport maneuvers. Possible values are:

- "publicTransport"
- "privateTransport"

Interface: Mode

This interface is a member of [nokia.maps.routing](#).

Interface Summary

This interface defines route modes, types and other options applicable to route calculations.

[For full details, see [nokia.maps.routing.Mode](#)]

Table 244: Property Summary

Properties
options: {String[]} [optional] This property specifies the routing options to be applied when calculating the route.
trafficMode: {String} [optional] This property indicates if traffic information is to be considered when calculating the route.
transportModes: {String[]} This property defines the transport mode to be used in the route calculation.
type: {String} This property holds a route type applicable to the route calculation.

here

Interface Description

This interface defines route modes, types and other options applicable to route calculations.

Property Details

options: {String[]} [optional]

This property specifies the routing options to be applied when calculating the route. The possible values are:

- "avoidTollroad" - indicates that the route should avoid toll roads
- "avoidMotorway" - indicates that the route should avoid motorways (highways)
- "avoidBoatFerry" - indicates that the route should avoid boat ferries
- "avoidRailFerry" - indicates that the route should avoid rail ferries
- "avoidPublicTransport" - indicates that the route should avoid public transport
- "avoidTunnel" - indicates that the route should avoid tunnels
- "avoidDirtRoad" - indicates that the route should avoid dirt roads
- "avoidPark" - indicates that the route should avoid parks
- "preferHOVLane" - indicates that the route should give preference to high-occupancy vehicle (HOV) lanes
- "avoidStairs" - indicates that the route should avoid stairs

trafficMode: {String} [optional]

This property indicates if traffic information is to be considered when calculating the route. The possible values are:

- "enabled" - indicates that the dynamic traffic conditions are to be taken into account (current traffic traffic patterns, short term closures, long term closures)
- "disabled" - indicates that dynamic traffic conditions are not to be taken into account, but only time restrictions and seasonal closures
- "default" - indicates that the service is to apply traffic-related constraints suitable for the selected routing type, transport mode and departure time, and it is to take into consideration user entitlements automatically

transportModes: {String[]}

This property defines the transport mode to be used in the route calculation. The possible values are:

here

- "car" - indicates that the route is to be suitable for cars
- "pedestrian" - indicates that the route is for pedestrians (walking)
- "truck" - indicates that the route is to be suitable for trucks (heavy goods vehicles)

type: {String}

This property holds a route type applicable to the route calculation. The possible values are:

- "shortest" - the shortest route between the start and finish points, disregarding traffic conditions
- "fastest" - a route that allows the destination to be reached in the shortest amount of time; the travel time takes into consideration traffic information (if `trafficMode` is selected)
- "fastestNow" - a convenience option that combines the route type "fastest" with the traffic mode "enabled" and takes into consideration the departure time; if the departure time is omitted, the current time ("now") is assumed
- "directDrive" - an option that combines the route type "fastest" and the traffic mode "disabled" and takes into consideration the departure time; if departure time is omitted, the current time ("now") is assumed
- "scenic" - a route that favors interesting scenery and landscapes

Interface: Route

This interface is a member of [nokia.maps.routing](#).

Interface Summary

This interface defines a route.

[For full details, see [nokia.maps.routing.Route](#)]

Table 245: Property Summary

Properties
legs: {nokia.maps.routing.RouteLeg[]} This property holds an array that represents the partition of the route into legs between the different waypoints.
mode: {nokia.maps.routing.Mode} This property holds an object that represents the route mode used in the route calculation.
shape: {nokia.maps.geo.Shape} This property holds a route shape object as a polyline.

here

Properties

summary: {*nokia.maps.routing.RouteSummary*}

This property holds information about the overall route distance and time summary.

waypoints: {*nokia.maps.routing.Waypoint*[]}

This property holds a list of waypoints that have been defined when requesting a route calculation.

Interface Description

This interface defines a route. A route describes a path through the route network between at least two waypoints (start and end). It is calculated by a back-end server and includes a list of route links and maneuvers. A maneuver describes an action a person needs to perform to follow a route. For example, it can contain an instruction to turn right or left or an indication of the direction of travel, etc. A maneuver applies to a route link, which is a part of a route that has a unique id (allocated by the server) and consists of a list of points/locations (defined in terms of their geographic coordinates).

Property Details

legs: {*nokia.maps.routing.RouteLeg*[]}

This property holds an array that represents the partition of the route into legs between the different waypoints.

mode: {*nokia.maps.routing.Mode*}

This property holds an object that represents the route mode used in the route calculation.

shape: {*nokia.maps.geo.Shape*}

This property holds a route shape object as a polyline. Its accuracy depends on the resolution specified in mpp (meters per pixel) when requesting the route. In some use cases (such as Web portals), only the route's shape is required, without the nested structure and detailed knowledge of the links and their ids; the shape does not need to be acquired by traversing the route's links, but can be represented using this attribute at route level.

summary: {*nokia.maps.routing.RouteSummary*}

This property holds information about the overall route distance and time summary.

here

waypoints: {[nokia.maps.routing.Waypoint](#)[]}}

This property holds a list of waypoints that have been defined when requesting a route calculation. The first waypoint is defined as the start of the route; the last waypoint marks the destination. Any waypoints in between the two define "via points".

Interface: RouteLeg

This interface is a member of [nokia.maps.routing](#).

Interface Summary

This interface defines a route leg, which represents a section of a route between two consecutive waypoints.

[For full details, see [nokia.maps.routing.RouteLeg](#)]

Table 246: Property Summary

Properties
end: {nokia.maps.routing.Waypoint} This property holds an object representing the waypoint at the end of the route leg.
length: {Number} This property holds the length of the given leg of the route.
link: {String[]} List of all links which are included in this portion of the route.
maneuvers: {nokia.maps.routing.Maneuver[]}} This property holds a list of all maneuvers which are included in this section of the route.
start: {nokia.maps.routing.Waypoint} This property holds an object representing the waypoint at the start of the route leg.
subLegSummary: {nokia.maps.advrouting.RouteSummary} Distance and time summary information for any sub legs of this route leg.
travelTime: {Number} This property holds the time needed to travel the length of the given route leg.

Interface Description

This interface defines a route leg, which represents a section of a route between two consecutive waypoints.

here

Property Details

end: {*nokia.maps.routing.Waypoint*}

This property holds an object representing the waypoint at the end of the route leg. It refers to one of the waypoints in the Route object, an element in Route.waypoints.

length: {Number}

This property holds the length of the given leg of the route.

link: {String[]}

List of all links which are included in this portion of the route.

maneuvers: {*nokia.maps.routing.Maneuver*[]}

This property holds a list of all maneuvers which are included in this section of the route.

start: {*nokia.maps.routing.Waypoint*}

This property holds an object representing the waypoint at the start of the route leg. It refers to one of the waypoints in the Route object, an element in Route.waypoints.

subLegSummary: {*nokia.maps.advrouting.RouteSummary*}

Distance and time summary information for any sub legs of this route leg. The service defines sub legs where passThrough waypoints are used, so the list may be empty if no such waypoints exist within this route leg.

travelTime: {Number}

This property holds the time needed to travel the length of the given route leg. The routing type determines if this value takes into account traffic information.

Interface: RouteSummary

This interface is a member of *nokia.maps.routing*.

here

Interface Summary

This interface defines a route summary.

[For full details, see [nokia.maps.routing.RouteSummary](https://developers.google.com/maps/documentation/javascript/route-api#RouteSummary)]

Table 247: Property Summary

Properties
<i>baseTime</i>: {Number} This property holds the total travel time required to travel the length of the route, not considering traffic, but taking the transport mode into account.
<i>distance</i>: {Number} This property holds the total distance covered by the route (the distance a traveler covers by following the route from start to finish).
<i>flags</i>: {String[]} This property contains an array of strings that hold special criteria that specify, for example, ferry or motorway usage, and which are matched by the route.
<i>trafficTime</i>: {Number} This property holds the total time required to travel the length of the route, considering traffic.

Interface Description

This interface defines a route summary.

Property Details

baseTime: {Number}

This property holds the total travel time required to travel the length of the route, not considering traffic, but taking the transport mode into account.

distance: {Number}

This property holds the total distance covered by the route (the distance a traveler covers by following the route from start to finish).

flags: {String[]}

This property contains an array of strings that hold special criteria that specify, for example, ferry or motorway usage, and which are matched by the route. The possible values are:

here

- "motorway" - indicates that the link is a motorway
- "boatFerry" - indicates that the link can only be traversed by using a boat ferry
- "railFerry" - indicates that the link can only be traversed by using a rail ferry
- "publicTransport" - indicates that the link can only be traversed by using public transport
- "gatedArea" - indicates that the link is part of a gated area
- "privateRoad" - indicates that the link is part of a private road
- "tollroad" - indicates that the link is part of a toll road
- "station" - indicates a public transport station

trafficTime: {Number}

This property holds the total time required to travel the length of the route, considering traffic.

Interface: ServiceError

This interface is a member of [nokia.maps.routing](#).

Interface Summary

This interface represents a routing service error.

[For full details, see [nokia.maps.routing.ServiceError](#)]

Table 248: Property Summary

Properties
message: {String} This property contains the error message with textual explanation of the error.
subtype: {String} This property holds a subtype of the service error.
type: {String} This property holds the service error type.

Interface Description

This interface represents a routing service error.

here

Property Details

message: {String}

This property contains the error message with textual explanation of the error.

subtype: {String}

This property holds a subtype of the service error. The possible values are:

- "invalidInputData" - indicates that the request contains invalid input data (wrong format, contradictory input, etc.)
- "contractViolate" - indicates that the contract exists for the given token, but is not yet active or has already expired
- "permissionError" - indicates that the access permission for the given token has been denied by the service
- "exceededUsageLimit" - indicates that the usage limit as given in the user's contract associated with the given token has been reached
- "invalidCredentials" - indicates that the specified token was invalid or no contract could be found for this token
- "insufficientRights" - indicates that the user's contract exists for the given token, but a required entitlement is missing
- "networkAccessDenied" - indicates that network access to the requested resource has been denied
- "serviceNotAvailable" - indicates that the service is not available (server down, resource not found, etc.)
- "timeout" - indicates that a timeout occurred
- "noRouteFound" - indicates that no route could be constructed based on the input parameter(s)
- "waypointNotFound" - indicates that one of the requested way points (start/end or a via point) could not be found in the routing network
- "linkIdNotFound" - indicates that a link id passed as input parameter could not be found in the underlying map data
- "invalidRouteId" - indicates that the route could not be reconstructed based on the given route id
- "positionOffRoute" - indicates that the specified position is too far away from the route; a position is considered to be off the route if it lies 500m or more away from it

here

type: {String}

This property holds the service error type. The possible values are:

- "applicationError" - indicates an error, which was thrown because the business logic has detected some error
- "systemError" - indicates an error, which was thrown due to technical reasons
- "permissionError" - indicates an error, which was thrown due to invalid credentials or missing entitlements

Interface: Waypoint

This interface is a member of [nokia.maps.routing](#).

Interface Summary

This interface defines a waypoint, which is a point along a route explicitly specified as part of route definition as a location through which the route must pass.

[For full details, see [nokia.maps.routing.Waypoint](#)]

Table 249: Property Summary

Properties
mappedPosition: {nokia.maps.geo.Coordinate} This property holds an object containing the geographical coordinates of the nearest point on the link (route) to the original position specified as part of the route calculation request (the position of the waypoint on the calculated route).
originalPosition: {nokia.maps.geo.Coordinate} This property holds an object containing the geographical coordinates of the position of the waypoint as it was originally specified in the route calculation request.

Interface Description

This interface defines a waypoint, which is a point along a route explicitly specified as part of route definition as a location through which the route must pass. A waypoint can correspond to a route link position ([nokia.maps.advrouting.LinkPosition](#)) or it can be the result of map matching. In the first case, the attribute mappedPosition is not filled.

Property Details

mappedPosition: {[nokia.maps.geo.Coordinate](#)}

here

This property holds an object containing the geographical coordinates of the nearest point on the link (route) to the original position specified as part of the route calculation request (the position of the waypoint on the calculated route).

originalPosition: {[*nokia.maps.geo.Coordinate*](#)}

This property holds an object containing the geographical coordinates of the position of the waypoint as it was originally specified in the route calculation request.

Class: WaypointParameterList

This class is a member of [*nokia.maps.routing*](#).

Class Summary

This class represents a list of waypoints that may be defined by instances of various location classes.

[For full details, see [*nokia.maps.routing.WaypointParameterList*](#)]

Table 250: Method Summary

Methods
addCoordinate (coord) This method adds an object containing geographical coordinates to the given instance of WaypointParameterList.
addLocation (location) This method adds a location object to the given instance of WaypointParameterList.
addPlace (place) This method adds a place object to the given instance of WaypointParameterList.
clear () This method clears the given instance of the waypoint parameter list.
size () : {Number} This method retrieves the size of the given instance of the waypoint parameter list.
toCoordinates () : {<i>nokia.maps.geo.Coordinate</i>[]} This method converts the given instance of the WaypointParameterList to an array of instances of <i>nokia.maps.geo.Coordinate</i>.

here

Class Description

This class represents a list of waypoints that may be defined by instances of various location classes. Thus a waypoint may be specified, using an instance of [nokia.maps.geo.Coordinate](#), [nokia.maps.search.Place](#) or [nokia.maps.search.Location](#).

Constructor Details

nokia.maps.routing.WaypointParameterList()

This method initializes an instance of `WaypointParameterList` that contains a list of waypoints with potentially different representations.

Method Details

addCoordinate(coord)

This method adds an object containing geographical coordinates to the given instance of `WaypointParameterList`.

Parameters:

coord: {[nokia.maps.geo.Coordinate](#)}

An object containing the geographical coordinates of a location

addLocation(location)

This method adds a location object to the given instance of `WaypointParameterList`.

Parameters:

location: {[nokia.maps.search.Location](#)}

A location object to be added

addPlace(place)

This method adds a place object to the given instance of `WaypointParameterList`.

Parameters:

place: {[nokia.maps.search.Place](#)}

here

A place object to be added

clear()

This method clears the given instance of the waypoint parameter list.

size(): {Number}

This method retrieves the size of the given instance of the waypoint parameter list.

Returns:

{Number} A numeric value representing the number of waypoint parameters in the list

toCoordinates(): {*nokia.maps.geo.Coordinate*[]}

This method converts the given instance of the WaypointParameterList to an array of instances of *nokia.maps.geo.Coordinate*.

Returns:

{*nokia.maps.geo.Coordinate*[]}
An array of latitude-longitude objects

Namespace: component

This namespace is a member of *nokia.maps.routing*.

Namespace Summary

This namespace defines facilities that add interactive behavior to routes displayed on the map.

Namespace Description

This namespace defines facilities that add interactive behavior to routes displayed on the map.

Class: *IsolineResultSet*

This class is a member of *nokia.maps.routing.component*.

here

Class Summary

This class encapsulates a result set generated in response to a request for an isoline routing calculation.

[For full details, see [nokia.maps.routing.component.IsolineResultSet](#)]

Table 251: Property Summary

Properties
container: { nokia.maps.map.Container }
This property is a container for the isoline, representing the results of the isoline routing calculation.

Class Description

This class encapsulates a result set generated in response to a request for an isoline routing calculation. The class provides methods to handle the response data in combination with a map.

Constructor Details

nokia.maps.routing.component.IsolineResultSet(isoline, settings)

This method initializes an instance of an IsolineResultSet with a response to a request for an isoline routing calculation.

Parameters:

- isoline:

{[nokia.maps.geo.Shape](#)}

An object containing the isoline data which represent the result of an isoline routing calculation.
- settings:

{Object}

An optional hash object with settings for the drawing of the isoline. The possible attributes of the object are:
 - "brush" - a brush object used to fill the isoline
 - "pen" - a pen object used to draw the border of the isoline
- For further details of the attributes, please see [nokia.maps.map.Polygon.Properties](#)

here

Property Details

container: {[*nokia.maps.map.Container*](#)}

This property is a container for the isoline, representing the results of the isoline routing calculation.

Class: *RouteResultSet*

This class is a member of [*nokia.maps.routing.component*](#).

Class Summary

This class represents a route calculation result set.

[For full details, see [*nokia.maps.routing.component.RouteResultSet*](#)]

Class Description

This class represents a route calculation result set. It can be initiated with a response from a route calculation request and provides methods to handle this response in combination with a map.

Constructor Details

nokia.maps.routing.component.RouteResultSet(routedata, settings)

The constructor initializes an instance of *RouteResultSet* on the basis of the arguments supplied by the caller. A new instance can be created, using a response to a route calculation request.

Parameters:

routedata: {[*nokia.maps.routing.Route*](#)}

The route data which represent the results of a route calculation

settings: {Object}

An optional hash object specifying settings to be used when drawing a route; the possible attributes are as defined under [*nokia.maps.map.Polyline.Properties*](#) and include:

- "pen" - an object that defines the properties of a polyline drawn on the map (see also [*nokia.maps.util.Pen*](#))
- "arrows" - an object that defines the properties of arrows drawn on the map (see also [*nokia.maps.util.Arrows*](#))

here

Namespace: search

This namespace is a member of [nokia.maps](#).

Namespace Summary

This namespace provides search, including places search, as well as geocoding and reverse geocoding functionality.

Namespace Description

This namespace provides search, including places search, as well as geocoding and reverse geocoding functionality.

Interface: Address

This interface is a member of [nokia.maps.search](#).

Interface Summary

This interface defines a location address, for example a street address.

[For full details, see [nokia.maps.search.Address](#)]

Table 252: Property Summary

Properties
city : {String} This property refers to the locality of the address.
country : {String} This property holds an ISO 3166-alpha-3 country code.
county : {String} This property includes the second subdivision level(s) below the country.
district : {String} This property holds the subdivision level below the city.
houseNumber : {String} This property holds the house number.
label : {String}

here

Properties

This property holds an assembled address for display purposes.

postalCode: {String}

This property is designed to hold the postal code.

state: {String}

This property includes the first subdivision level(s) below the country.

street: {String}

This property holds the street name in the requested primary language.

Interface Description

This interface defines a location address, for example a street address. The attributes are normalized to US postal address property names that can be mapped to localized address properties, using mapping tables for GDF (for example, "State" matches "Bundesland" in Germany).

Property Details

city: {String}

This property refers to the locality of the address. Some products map "city" directly to Address.city. For GDF, a country-specific mapping is required.

country: {String}

This property holds an ISO 3166-alpha-3 country code.

county: {String}

This property includes the second subdivision level(s) below the country. The use of this field is optional if a second subdivision level is not available.

district: {String}

This property holds the subdivision level below the city. The use of this field is optional if a second subdivision level is not available.

houseNumber: {String}

here

This property holds the house number. Depending on regional characteristics, the value may also be a house name. For API implementations which do not support separate street and house number fields, this field can be omitted.

label: {String}

This property holds an assembled address for display purposes.

postalCode: {String}

This property is designed to hold the postal code.

state: {String}

This property includes the first subdivision level(s) below the country. Some products map "region" directly to Address.state. For GDF, a country-specific mapping is required.

street: {String}

This property holds the street name in the requested primary language. For API implementations which do not support separate street and house number fields, this field may also contain the house number value.

Interface: Location

This interface is a member of [nokia.maps.search](#).

Interface Summary

This interface defines a location.

[For full details, see [nokia.maps.search.Location](#)]

Table 253: Property Summary

Properties
address: {nokia.maps.search.Address} This property holds a reference to an optional Address object.
displayPosition: {nokia.maps.geo.Coordinate} This property holds an object containing the geographic coordinates of the location, where map markers should be placed.

here

Properties

label: {String}

This property holds a descriptive label of the location for display purposes in the requested primary language.

mapView: {*nokia.maps.geo.BoundingBox*}

This property holds a bounding box which completely encloses the physical extent of the location.

navigationPositions: {*nokia.maps.geo.Coordinate*[]}

This property holds an array of instances of *nokia.maps.geo.Coordinate* That represent the location when it is used for routing waypoints.

Interface Description

This interface defines a location. The location type refers to a physical location including its physical extent. A location can be referenced either by LRO ID or by its address.

Property Details

address: {*nokia.maps.search.Address*}

This property holds a reference to an optional Address object.

displayPosition: {*nokia.maps.geo.Coordinate*}

This property holds an object containing the geographic coordinates of the location, where map markers should be placed.

label: {String}

This property holds a descriptive label of the location for display purposes in the requested primary language.

mapView: {*nokia.maps.geo.BoundingBox*}

This property holds a bounding box which completely encloses the physical extent of the location.

navigationPositions: {*nokia.maps.geo.Coordinate*[]}

This property holds an array of instances of *nokia.maps.geo.Coordinate* That represent the location when it is used for routing waypoints.

here

Interface: LocationFilter

This interface is a member of [nokia.maps.search](#).

Interface Summary

This interface defines a location filter.

[For full details, see [nokia.maps.search.LocationFilter](#)]

Table 254: Property Summary

Properties
city : {String} This property holds the identifier of the administrative area at GDF Order8 level (examples US: City, DE: Gemeinde).
country : {String} This property holds the country code (3 bytes, ISO 3166-1-alpha-3).
county : {String} This property holds the identifier of the administrative area at GDF Order2 level (examples US: County, DE: Kreis)
district : {String} This property holds the identifier of the administrative area at GDF Builtup level (examples US: n/a, DE: Ortsteil).
houseNumber : {String} This property holds the house number.
locationId : {String} This property holds the location id, which uniquely identifies a physical location within the LRO system.
postalCode : {String} This property holds the postal code.
state : {String} This property holds the identifier of the administrative area at GDF Order1 level (examples US: State, DE: Bundesland).
street : {String String[]} This property holds the name of the street.

Interface Description

This interface defines a location filter. A filter is designed to match search results to certain address values or to match an exact location reference (its LRO ID).

here

Property Details

city: {String}

This property holds the identifier of the administrative area at GDF Order8 level (examples US: City, DE: Gemeinde). It can also be an alternative administrative name (e.g., "Paris" for Charles-de-Gaulle airport, which is actually located in Roissy-en-France).

country: {String}

This property holds the country code (3 bytes, ISO 3166-1-alpha-3).

county: {String}

This property holds the identifier of the administrative area at GDF Order2 level (examples US: County, DE: Kreis)

district: {String}

This property holds the identifier of the administrative area at GDF Builtup level (examples US: n/a, DE: Ortsteil).

houseNumber: {String}

This property holds the house number. Depending on regional characteristics, the property value can also be a house name.

locationId: {String}

This property holds the location id, which uniquely identifies a physical location within the LRO system.

postalCode: {String}

This property holds the postal code.

state: {String}

here

This property holds the identifier of the administrative area at GDF Order1 level (examples US: State, DE: Bundesland).

street: {String | String[]}

This property holds the name of the street. A second street name can be included to search for street intersections.

Class: Manager

This class is a member of [nokia.maps.search](#).

Extends: [nokia.maps.util.OObject](#)

Class Summary

This class acts as a proxy to the search service and provides methods for retrieving the search results.

[For full details, see [nokia.maps.search.Manager](#)]

Table 255: Property Summary

Properties
gen: {Number} The gen parameter enables or disables backward incompatible behavior in <code>geocode()</code> and <code>reverseGeocode()</code>.
locations: {nokia.maps.search.Location[]} This property holds a read-only array of locations which were returned by a query.
maxResults: {Number} This property represents the maximum number of search results to be returned by calls to <code>geocode()</code> or <code>reverseGeocode()</code>.
places: {nokia.maps.search.Place[]} This property holds a read-only array of places which were returned by a query.
state: {String} This property is a read-only field that signals state changes in the manager.

Table 256: Method Summary

Methods
clear ()

here

Methods

This method resets the stored search request and response.

geocode (geocodeItem, spatialFilter)

This method performs geocoding, using the back-end search service.

getErrorCause () : {*nokia.maps.search.ServiceError*}

This method retrieves an object containing the search error code and associated error message text.

reverseGeocode (position)

This method submits a reverse geocoding request based on the location specified by the caller.

Directly Inherited Methods

Inherited from class *nokia.maps.util.OObject* :

addObserver, *get*, *remove*, *removeObserver*, *set*

Class Description

This class acts as a proxy to the search service and provides methods for retrieving the search results. We assume that only one search response is kept at a time. If a new request is triggered, the old response is replaced.

Constructor Details

nokia.maps.search.Manager ()

This constructor initializes the client-side interface of the search functionality. We assume that only one search response is kept at a time. If a new request is triggered, the old response is replaced.

Property Details

gen: {Number}

The gen parameter enables or disables backward incompatible behavior in *geocode* () and *reverseGeocode* () .

- gen=3 (default)
 - Includes gen=2 functionality.
 - Reverse geocoding address results have display and navigation positions. With gen<3 address results from reverse geocoding only have a display position which in actuality is the navigation position. With gen=3 this bug is fixed.

here

- MatchType pointAddress for reverse geocoding Point Address results. Otherwise interpolated. With gen<3 address results from reverse geocoding always have MatchType interpolated.
- gen=2
 - Includes gen=1 functionality.
 - Return the area display point (e.g. the centroid of the area) for reverse geocode area responses. With gen<2 the DisplayPoint simply mirrors the point from the request.
 - Return map version in element MapReference if requested via locationattributes=(one of mr, mapReference, all)
- gen=1
 - If a reverse geocoding request for an address does not find an address or street within the set radius the area information is returned instead. If gen=0 the results with a 10,000 meter radius are returned.
 - Result MatchLevel for intersection matches is intersection. With gen=0 the MatchLevel for intersection matches is street.
- gen=0 original behavior

Default Value: 3

locations: {*nokia.maps.search.Location*[][]}

This property holds a read-only array of locations which were returned by a query.

maxResults: {Number}

This property represents the maximum number of search results to be returned by calls to `geocode()` or `reverseGeocode()`.

places: {*nokia.maps.search.Place*[][]}

This property holds a read-only array of places which were returned by a query. Places can contain multiple sub locations. Every sub location is also contained in the overall location property.

state: {String}

here

This property is a read-only field that signals state changes in the manager. The possible values are: "initial", "started", "finished", "failed" and "cleared".

Method Details

clear()

This method resets the stored search request and response.

geocode(geocodeItem, spatialFilter)

This method performs geocoding, using the back-end search service.

Parameters:

geocodeItem: {String | *nokia.maps.search.LocationFilter*}

A string that contains search text (unqualified search) or a location filter object (qualified address search)

spatialFilter: {*nokia.maps.geo.BoundingBox* | *nokia.maps.geo.Corridor* | *nokia.maps.geo.Proximity*}

A bounding box, a corridor or a proximity object to be used to limit the result set to a certain geographic area

getErrorCause(): {*nokia.maps.search.ServiceError*}

This method retrieves an object containing the search error code and associated error message text.

Returns:

{*nokia.maps.search.ServiceError*}

An object containing information about the search error

reverseGeocode(position)

This method submits a reverse geocoding request based on the location specified by the caller. A default radius is applied.

Parameters:

here

position: {*nokia.maps.geo.Coordinate*}

An object containing the geo coordinates of a location that represents the input for the reverse geocoding request

Interface: Place

This interface is a member of *nokia.maps.search*.

Interface Summary

This interface defines a place object and its properties.

[For full details, see *nokia.maps.search.Place*]

Table 257: Property Summary

Properties
<i>categories</i> : {String[]} This property holds an array of strings which represent a list of POI category names.
<i>locations</i> : { <i>nokia.maps.search.Location</i> []} This property holds an array of Location objects.
<i>name</i> : {String} This property holds the descriptive name of a place in the requested primary language for display purposes.
<i>primaryEmail</i> : This property holds the primary e-mail address of the place.
<i>primaryFax</i> : This property holds the primary fax number of the place.
<i>primaryIM</i> : This property holds the primary instant messaging account of the place.
<i>primaryPhone</i> : This property holds the primary phone number of the place.
<i>primaryURL</i> : This property holds the primary URL of the place.
<i>suppliers</i> : {String[]} This property holds an array of strings containing supplier names.

here

Interface Description

This interface defines a place object and its properties. A place is an entity that encapsulates information about objects, services, etc., that can be found at a specific location. A place can represent a single point defined its latitude and longitude or it may be an area or feature of the landscape identified as a Point of Interest (POI). Information often associated with a place includes contact details and media (images and reviews).

Property Details

categories: {String[]}

This property holds an array of strings which represent a list of POI category names.

locations: {*nokia.maps.search.Location*[]}

This property holds an array of *Location* objects.

name: {String}

This property holds the descriptive name of a place in the requested primary language for display purposes.

primaryEmail:

This property holds the primary e-mail address of the place.

primaryFax:

This property holds the primary fax number of the place.

primaryIM:

This property holds the primary instant messaging account of the place.

primaryPhone:

This property holds the primary phone number of the place.

here

primaryURL:

This property holds the primary URL of the place.

suppliers: {String[]}

This property holds an array of strings containing supplier names.

Interface: ServiceError

This interface is a member of [nokia.maps.search](#).

Interface Summary

This interface represents a search service error.

[For full details, see [nokia.maps.search.ServiceError](#)]

Table 258: Property Summary

Properties
message: {String} This property holds additional information about the error.
subtype: {String} This property holds the subtype of the service error.
type: {String} This property holds the identifier of the service error type.

Interface Description

This interface represents a search service error.

Property Details

message: {String}

This property holds additional information about the error.

subtype: {String}

This property holds the subtype of the service error. The possible values are:

here

- "invalidInputData" - indicates that the request contains invalid input data (wrong format, contradictory input, etc.)
- "contractViolate" - indicates that the contract exists for given token, but is not yet active or has already expired
- "permissionError" - indicates that the access permission for the given token has been denied by the service
- "exceededUsageLimit" - indicates that the usage limit specified in the user's contract which is assigned to the given token has been reached
- "invalidCredentials" - indicates that the specified token was invalid or no contract could be found for this token
- "insufficientRights" - indicates that the user's contract exists for the given token, but a required entitlement is missing
- "networkAccessDenied" - indicates that network access to the requested resource has been denied
- "serviceNotAvailable" - indicates that the service is not available (server down, resource not found, etc.)
- "timeout" - indicates that a time-out occurred

type: {String}

This property holds the identifier of the service error type. The possible values are:

- "applicationError" - indicates an error thrown because the business logic has detected some error
- "systemError" - indicates a technical error
- "permissionError" - indicates an error thrown due to invalid credentials or missing entitlements

Namespace: component

This namespace is a member of [*nokia.maps.search*](#).

Namespace Summary

This namespace defines facilities to help make search functionality interactive.

Namespace Description

This namespace defines facilities to help make search functionality interactive.

here

Class: *GeocodeComponent*

This class is a member of [nokia.maps.search.component](#).

Extends: [nokia.maps.map.component.Component](#)

Class Summary

This class shows the results of a geocoding on the display.

[For full details, see [nokia.maps.search.component.GeocodeComponent](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

[mapDisplay](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :

[attach](#), [destroy](#), [detach](#), [getId](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This class shows the results of a geocoding on the display. To do this, it retrieves the results from an instance of [nokia.maps.search.component.SearchResultSet](#) that is available as consequence of a call to a method on [nokia.maps.search.Manager](#).

Constructor Details

nokia.maps.search.component.GeocodeComponent(searchManager)

This method initializes and instance of [nokia.maps.search.component.GeocodeComponent](#) to provide a means of displaying geocoding results.

Parameters:

here

searchManager: {[nokia.maps.search.Manager](#)}

An instance of a Search Manager.

Class: SearchComponent

This class is a member of [nokia.maps.search.component](#).

Extends: [nokia.maps.map.component.Component](#)

Class Summary

This class displays the results of a search.

[For full details, see [nokia.maps.search.component.SearchComponent](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.map.component.Component](#) :

[mapDisplay](#)

Method Summary

Directly Inherited Methods

Inherited from class [nokia.maps.map.component.Component](#) :

[attach](#), [destroy](#), [detach](#), [getId](#)

Inherited from class [nokia.maps.util.OObject](#) :

[addObserver](#), [get](#), [remove](#), [removeObserver](#), [set](#)

Class Description

This class displays the results of a search. To do this, it retrieves the search results from an instance of [nokia.maps.search.component.SearchResultSet](#) available as a result of calling `placeSearch()` on [nokia.maps.search.Manager](#).

Constructor Details

`nokia.maps.search.component.SearchComponent(searchManager)`

here

This method initializes an instance of `nokia.maps.search.component.SearchComponent` to make it possible to display search results.

Parameters:

`searchManager:` {[*nokia.maps.search.Manager*](#)}
An instance of search manager through which to run a search and obtain the search results

Class: SearchResultSet

This class is a member of [*nokia.maps.search.component*](#).

Class Summary

This class represents search results.

[For full details, see [*nokia.maps.search.component.SearchResultSet*](#)]

Table 259: Property Summary

Properties
<i>container:</i> { <i>nokia.maps.map.Container</i> } This property holds a container of markers, representing the results.

Class Description

This class represents search results. An instance of this class can be initiated with a response to a search request (method call on `searchManager` and provides some useful methods to handle this response in combination with a map.

Constructor Details

`nokia.maps.search.component.SearchResultSet`(searchResults)

This method initializes an instance of `SearchResultSet`, using the arguments provided by the caller.

Parameters:

`searchResults:` {[*nokia.maps.search.Location*](#)[]}
An array containing search results obtained via `search.Manager`

here

Property Details

container: {[nokia.maps.map.Container](#)}

This property holds a container of markers, representing the results.

Namespace: util

This namespace is a member of [nokia.maps](#).

Namespace Summary

This namespace defines utility classes and interfaces used in other namespaces in this API to support map objects, animations and graphics.

Namespace Description

This namespace defines utility classes and interfaces used in other namespaces in this API to support map objects, animations and graphics.

Class: Arrows

This class is a member of [nokia.maps.util](#).

Class Summary

This class defines the properties of arrows, used when stroking map polylines (routes) with arrows to indicate the direction of travel.

[For full details, see [nokia.maps.util.Arrows](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.util.IArrows](#) :

[color](#), [frequency](#), [length](#), [width](#)

Class Description

This class defines the properties of arrows, used when stroking map polylines (routes) with arrows to indicate the direction of travel.

here

Constructor Details

nokia.maps.util.Arrows(props)

This method initializes an instance of **Arrows**. Note that if the caller does not provide the initial property values, the method uses defaults.

Parameters:

props: {[nokia.maps.util.IArrows](#)}

A JavaScript object containing the initial values of the **Arrows** properties, for example, {"frequency": 4, "length": 2.3, "width": 1.0, "color": "#FFFC"}

Class: Brush

This class is a member of [nokia.maps.util](#).

Class Summary

This class defines the properties of brush objects that can be used when filling map shapes with color.

[For full details, see [nokia.maps.util.Brush](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.util.IBrush](#) :

[color](#), [fill](#)

Class Description

This class defines the properties of brush objects that can be used when filling map shapes with color.

Constructor Details

nokia.maps.util.Brush(props, brush)

This method initializes an instance of **Brush**. Note that if the caller does not provide initial property values, the method uses defaults.

here

Parameters:

- props:

`{nokia.maps.util.IBrush}`
The initial values of the Brush properties as a JavaScript object
- brush:

`{nokia.maps.util.Brush}` [optional]
If provided, this object is copied to create a new instance of the class

Class: Cache

This class is a member of [nokia.maps.util](#).

Class Summary

A cascadable LRU-Cache (Least Recently Used).

[For full details, see [nokia.maps.util.Cache](#)]

Table 260: Property Summary

Properties
<code>change</code>: A method whose return value should be returned by <code>addFilter</code> function to signal "The entry data has been changed by <code>addFilter</code>".
<code>static change</code>: A method whose return value should be returned by <code>addFilter</code> function to signal "The entry data has been changed by <code>addFilter</code>".
<code>static constant DONT</code>: {Object} The value to return by <code>addFilter</code> function to signal "The entry should not be added".
<code>static constant NEXT</code>: {Object} The value to return by <code>addFilter</code> function to signal "The entry should be added in next level if possible".
<code>static constant NOT_ENOUGH_SPACE</code>: {Object} A return value of <code>nokia.maps.util.Cache#add</code> to signal that the cache can't store the data because there is not enough space to store the data.

Table 261: Method Summary

Methods
<code>add</code> (<code>id</code>, <code>data</code>, <code>size</code>) : {Object undefined} To add an entry to the cache.

here

Methods

`destroy` ()

To clean up the cache on shut down.

`get` (id, noUpdate) : {Variant}

To get a possibly cached entry.

`remove` (id, nonRecursive) : {Variant}

To remove a possibly existing entry from cache.

`removeAll` (idPrefix)

To remove all cache entries which starts with the given id prefix.

Class Description

A cascable LRU-Cache (Least Recently Used). Each cache level is configurable by it's size plus optional addFilter and an onDrop callback.

Constructor Details

`nokia.maps.util.Cache`(size, addFilter, onDrop, nextCache)

Parameters:

size: {Number} [optional, default: 1]

The maximum size of this this cache for all entry data. The size must be a number between 0...Infinity. See also [nokia.maps.util.Cache#add](#) size argument.

addFilter: {Function} [optional]

A filter to use when a cache entry should be added via add(). If the addFilter argument is omitted the original data will be stored. This function will get the data of the cache entry to add as argument. It has to return one of the following values:

- [nokia.maps.util.Cache.DONT](#) to signal that the entry should not be added and possibly existing entries should be removed.
- [nokia.maps.util.Cache.NEXT](#) to signal that the entry should be added in next cache level.

here

- The return value of [nokia.maps.util.Cache.change](#) to signal that the entry has been changed.

onDrop: {Function} [optional]

A call-back which will be triggered if a cache entry dropped out on the last level. This function will get the data of the dropped cache entry as argument. This call-back can be used to clean-up cache entry data.

nextCache: {[nokia.maps.util.Cache](#)} [optional]

A cache which will used as next level. Least recently used entries of this cache will automatically moved the the nextCache when the maximum size of this cache is reached. If omitted the least recently used entry will be dropped instead.

Property Details

change:

A method whose return value should be returned by addFilter function to signal "The entry data has been changed by addFilter".

static change:

A method whose return value should be returned by addFilter function to signal "The entry data has been changed by addFilter".

static constant DONT: {Object}

The value to return by addFilter function to signal "The entry should not be added".

static constant NEXT: {Object}

The value to return by addFilter function to signal "The entry should be added in next level if possible".

static constant NOT_ENOUGH_SPACE: {Object}

A return value of [nokia.maps.util.Cache#add](#) to signal that the cache can't store the data because there is not enough space to store the data.

here

Method Details

add(id, data, size): {Object | undefined}

To add an entry to the cache.

Parameters:

id: {String}

The ID of the entry to add.

data: {Variant}

The data of the entry to cache.

size: {Number} [optional, default: 1]

The size of the data. If omitted or undefined the default value will be taken. See also [nokia.maps.util.Cache#add](#) size argument. It's recommended to use a rough estimation of the data size in byte. As an example: a string has a size of 2 * length (16 bit Unicode), a IMG has a size of width * height * 4 (RGBA), a number has a size of 8 Byte (IEEE double) etc.pp.

Returns:

{Object | undefined} the status of the add operation. The return value could be one of:

- undefined - if the entry was added to the cache
- [nokia.maps.util.Cache.NEXT](#) if the entry could not added because a addFilter has signaled "NEXT" but there is no next cache level.
- [nokia.maps.util.Cache.DONT](#) if the entry could not added because a addFilter has signaled "DONT".

destroy()

To clean up the cache on shut down.

get(id, noUpdate): {Variant}

To get a possibly cached entry.

Parameters:

here

id: {String}

The ID of the requested cache entry.

noUpdate: {Boolean} [optional, default: false]

A flag to signal that the cache should not be updated so that the requested entry will be not the "most recently used" entry.

Returns:

{Variant} the data of the cached entry or undefined if not found.

remove(id, nonRecursive): {Variant}

To remove a possibly existing entry from cache.

Parameters:

id: {String}

The ID of the entry to remove.

nonRecursive: {Boolean} [optional, default: false]

A flag to signal that the entry should NOT removed recursive in next cache levels.

Returns:

{Variant} The data of the removed entry or undefined if not found.

removeAll(idPrefix)

To remove all cache entries which starts with the given id prefix.

Parameters:

idPrefix: {String}

The ID prefix of the entries to remove.

here

Example:

```
myCache.removeAll(myProvider.id); // remove all cache entries whose ID
starts with the ID of myProvider
```

Class: Coroutine

This class is a member of [nokia.maps.util](#).

Class Summary

This class represents a coroutine that guarantees code execution in a non-blocking way.

[For full details, see [nokia.maps.util.Coroutine](#)]

Table 262: Property Summary

Properties
static AF_MAX_EXEC_TIME: {Number} This property holds the maximal execution time for those coroutine being executed within the animation frame request (see nokia.maps.util.Coroutine.ExecutionContext#useAnimationFrame and nokia.maps.util.Coroutine#MAX_EXEC_TIME for details.
static AF_SLICE_SIZE: {Number} This property holds the maximal size of each time slice for a coroutine being executed within the animation frame request (see nokia.maps.util.Coroutine.ExecutionContext#useAnimationFrame and nokia.maps.util.Coroutine#SLICE_SIZE for details.
static constant BLOCKED: {Object} This property holds a special object returned by the <code>sleep()</code>, <code>wait()</code> and <code>suspend()</code> methods.
static constant BLOCKING: {Number} Coroutine status constant indicating that a context is blocked at a condition or sleeping.
static constant KILLED: {Object} This property holds a special object returned by the <code>kill()</code> method.
static MAX_EXEC_TIME: {Number} This property holds the maximum execution time or the number of milliseconds consumed by coroutines before control is handed over to the browser and so to other JavaScript code.
static constant RUNNING: {Number} Coroutine status constant indicating that a context is currently running.
static SLICE_SIZE: {Number} This property holds the maximum size of each time slice for a coroutine.
static constant TERMINATED: {Number} Coroutine status constant indicating that a context is terminated.

here

Properties

static constant **YIELDED**: {Object}

This property holds a special object returned by the `yield()` method.

Table 263: Method Summary

Methods

static **all** () : {*nokia.maps.util.Coroutine.ExecutionContext*[]}}

This method returns an array with all the execution contexts currently marked as "running" or "blocking".

static **broadcast** (obj)

This method signals all coroutines blocking on an object and resumes them.

static **catchException** (enable)

This method enables exception catching in the scheduler to ignore errors in coroutines or it can be used to disable exception catching for debugging purposes.

static **count** () : {Number}

This method retrieves the number of contexts not being terminated, thus the return value includes all running and blocked contexts.

static **create** (name, fn, args) : {Function}

This method creates a new coroutine, using the name and implementation supplied by the caller.

static **createEx** (name, fn, defaults, args) : {Function}

This method creates a new coroutine using the name and implementation supplied by the caller.

static **current** () : {*nokia.maps.util.Coroutine.ExecutionContext*}

This method returns the currently executed context or null if no context is executed right now.

static **forId** (id) : {*nokia.maps.util.Coroutine.ExecutionContext*}

This method retrieves the execution context with the given id or undefined if no such context exists.

static **forName** (name) : {*nokia.maps.util.Coroutine.ExecutionContext*[]}}

This method searches through the list of all contexts currently in the "running" or "blocking" state and returns all those that are bound to the coroutine with the name supplied by the caller.

static **kill** (context, reason) : {Object}

This method immediately terminates the currently executed coroutine or the supplied coroutine context.

static **killAll** ()

This method destroys all execution contexts currently in the "running" or "blocking" state.

static **resume** (context)

This method resumes the supplied executed execution context.

static **scheduler** (isTestUnitCall)

here

Methods

This method forces the scheduler to run at once.

static `schedulerCpuTime` () : {Number}

This method retrieves the total CPU time that the scheduler itself has consumed.

static `schedulerNextRuntime` () : {Number}

This method retrieves the UNIX timestamp showing when the scheduler should run again.

static `scope` (fn)

This method clones the given function and moves the clone into the internal scope of the coroutine scheduler, removing the original scope remembered by the function.

static `shallYield` () : {Boolean}

This method checks if the currently executed coroutine context must yield or not.

static `signal` (obj) : {*nokia.maps.util.Coroutine.ExecutionContext*}

This method signals the given object and resumes the next coroutine that is blocked on this object, if any.

static `sleep` (timeout, context) : {Object}

This method puts a coroutine context to sleep.

static `suspend` (context, timeout) : {Object}

The method suspends the supplied execution context or the currently executed context.

static `taskCpuTime` () : {Number}

This method retrieves the total CPU time that all contexts have consumed.

static `totalCpuTime` () : {Number}

This method retrieves the number of milliseconds since the scheduler was started.

static `wait` (obj, timeout, context) : {Object}

This method blocks the a coroutine on the object provided by the caller.

static `yield` (context) : {Object}

This method makes the currently executed coroutine or the supplied one yield.

Class Description

There are many different ways to run code in a non-blocking way. The most common is to use threads which are parallel execution paths. Threads can be cooperative or preemptive. In a cooperative threading design, each thread or process is in full control of the CPU and must explicitly release the CPU so that another thread or process can take control of the CPU. In preemptive threading, a scheduler takes away the CPU from a thread and allocates it to another thread, therefore the thread has little or no control over when it is runs and on which CPU.

Preemptive versus cooperative threading

here

Each of the two threading models has certain advantages and disadvantages. Cooperative threading is much simpler for developers, because access control is not required. It guarantees that the memory (and so variables) and all other resources are only accessed by the currently executing code. It is not possible for one instruction to modify a variable, while another tries to read it or (worse) modify it too, therefore it impossible to access a value that is in a "half updated" state. This means that in cooperative threading there is no need for mutexes to synchronize access to memory (variables). Another advantage is performance. Because no synchronization is required, cooperative threading can be implemented more simply and a thread switch is much cheaper than in a preemptive environment.

However, the disadvantage of cooperative threading is that each process can only use one processor, because otherwise mutexes would become necessary, as each variable could be modified by one thread, while another is reading it (or even worse, both could be modifying the same object at the same time, possibly leaving the object in an unusable state). Therefore true parallel execution within one process is not possible. The other disadvantage is that a bug in the code may block the application, making it unresponsive. In extreme cases, it may block the entire operating system, making it unresponsive, because the thread executing the faulty code does not release the CPU.

JavaScript execution

All JavaScript code is executed in an execution context. Whenever some JavaScript code is executed, a new execution context is created. An execution context is an abstract collection that contains an activation and a variable object (as well internal information). These objects contain all the arguments supplied to a function and all the variables declared within the code, using the `var` statement. Because in reality the activation and variable objects are the same object, we simply use the term *variable object* to refer to both of them from here on. If an argument is named, its value is referenced in the *variable object* using that name; if a named argument is not supplied, a variable with the name of the argument exists in the *variable object*, but has the value `undefined` (the variable declared, but its value is `undefined`). This is important, because assigning a value to it does not lead to pollution of the global scope.

The scope chain

For any JavaScript code to be executed, there must be a global execution context in which this code can be run. In a browser environment, the browser creates such a global execution context and fills the *variable object* of that global context with some default values, for example it sets `document`, `navigator` or `Math`. The *variable object* in this global context contains a reference to the global context (`window`), which is helpful in modifying the global scope. So every browser window has its own global execution context, where the *variable object* of the global context can be accessed via the `window` variable.

To resolve a variable name to a value, the JavaScript interpreter needs to look up the variable name in the so-called *scope chain*. A newly created function "remembers" the current execution context.

here

Every time the function is called, the interpreter creates a new execution context and assigns the previous execution context as the parent of the new context, thus building the *scope chain*. If the JavaScript interpreter has to resolve a variable named "foo", it first searches in the *variable object* of the current execution context, which is the local context for the function. If no variable with that name is found there, the interpreter searches the parent execution context, then its parent, and so on, going up the chain until it reaches the global execution context (which has no parent). In short, the *scope chain* of a function is established at function create time and each subsequent call to the function adds a new element to that chain.

This is different for the `eval` statement, because `eval` executes the code that is passed to it in the current execution context. Therefore a variable declared within an `eval` statement with the `var` keyword extends the current execution context.

The following example shows the impact of this:

```
// Create new variables within the variable object
// of the current execution context.
//
var x = 1;
var y = 99;

// Create a new function -- it remembers the current execution
// context as its scope chain. The code is discussed below.
//
var alertX = function () {
    alert("x: "+x+", y: "+y);
    var y;
};

// Calling the function creates a new execution context with
// a new variable object, the parent execution context of this new
// execution context is set to the execution context that was
// remembered at the create time of the function.
//
// The interesting thing at this point is the output:
//
// "x: 1, y: undefined"
//
// The reason is that the new execution context contains a variable
// declaration for "y", but no value is assigned to it, while
// the only definition for "x" is in the parent context, and "x" is
// assigned a value there.
//
alertX();

// eval() is slow and should be avoided. The
// interpreter must evaluate the code passed to eval()
// within the current execution context, so the declaration of the
// variable "x" in that code has no effect, because "x" is already
// declared in the variable object of the previous execution context.
// Assigning 2 to "x" now modifies the "x" in the current execution
```

here

```
// context, therefore the following code causes alert()
// to display "2" twice.
//
eval("var x = 2; alert(x);");
alert(x);

// The following example demonstrates that the execution context is
// remembered
// at function create time. The example creates an anonymous function,
// remembers the current execution context and then directly calls
// the new function. The function call causes a new execution context to be
// created with the parent execution context set to the current
// execution context (note that create and call time of the function are
// the same now).
//
(function () {
  // The code below causes the variable with the name "x" to be declared
  // in the variable object of the current execution context. Therefore
  // the variable of parent context with the same name is no longer
  // accessible from within the function.
  //
  var x = 3;

  // However, the call to the function alertX() below, creates a new
  // execution
  // context with a new variable object, and assigns as the parent execution
  // context the execution context current at the time of the call. The
  // code
  // within alertX() cannot find a variable with the name "x" in its own
  // variable
  // object. It then searches its parent execution context (not the current
  // execution context, where it would find "x" set to 3). In the parent
  // context,
  // "x" has the value of 2. Thus, the following line outputs "2" and not
  // "3":
  //
  alertX();

  // Below, we create a new function. It is created there and then,
  // therefore it
  // remembers the current execution context. Note that it contains the same
  // code that the alertX() function.
  //
  var fn = function () {
    alert("x: "+x+", y: "+y);
    var y;
  };

  // So when we call this function, a new execution context is created
  // and the execution context remembered at the create time of the function
  // is assigned as the parent execution context. Because the create time
  // of this function is just one line above, it is the current execution
  // context. So the interpreter searches for "x" in the variable object
  // of the new execution context, not fails to find it and then searches
  // the parent
  // execution context, which is the current execution context that contains
```

here

```
// a variable with the name "x" and the value "3". Therefore this
// time, the output is "3" and not "2".
//
fn();
})();
```

Threading

The above-described *scope chain* explains why every execution context is bound to exactly one thread and every execution context must be bound to the thread to which its parent execution context is bound. For this reason, multi-threading is not possible in JavaScript and this is also why web-workers (the threads of HTML 5) do not have access to variables of the current window or to the DOM tree of the Web page. Every web-worker has its own global execution context and the execution context of a web-worker does not contain a reference to any window. That is also the reason why web-workers can only communicate using strings. Strings are immutable (read-only) and therefore there is no need to synchronize access to strings.

Most browsers use the same thread to render the UI, to parse the DOM tree and to execute JavaScript. Therefore as long as JavaScript is executed, no rendering of the UI or Web pages occurs, but note that this does not apply to all browsers. In any case, a buggy JavaScript code may block the browser and therefore all browser vendors implement some way to detect if JavaScript is blocking to prevent the browser becoming unresponsive to the user. For example, the Internet Explorer stops processing JavaScript after a certain number of instructions have been processed (by default around 5 million) and informs the user that a script is making the browser unresponsive, asking the user to continue the script or to abort it. Other browsers, for example Firefox, make the message conditional on elapsed execution time rather than a number of instructions.

Why you should use a coroutine and not `setTimeout()`

It can happen (and within this API it happens often) that some code must run for a number of seconds and/or more than 5 million instructions must be executed to perform a specific action. This happens, for example in map rendering to process huge polylines or a large number of polylines/polygons. Even though this does not always force an alert box of the browser to pop up, it makes the browser slow and unresponsive to the user input for as long as the JavaScript code runs. This situation may last a second or longer and it may occur multiple times in a row. Note that the performance of a browser is heavily dependent on the device on which the browser is running. Browsers do not have any built-in support for these cases, the function `setTimeout()` by itself does not help, because it does not take into account which function consumes how much processor time and so does not help the developer keep the browser responsive. It is possible to implement a function with a mechanism that prevents the browser becoming unresponsive, but problems are likely to arise if two or more such function run concurrently without any knowledge of one another. This API offers the [nokia.maps.util.Coroutine](#) class as a safe solution.

Support for coroutines

here

As a user of this API you can create your own coroutines. The API implements a scheduler ([nokia.maps.util.Coroutine.scheduler](#)) which keeps track of how much CPU time each coroutine consumes, ensuring that 80% of the available CPU time (so of the time not consumed by other JavaScript libraries) is distributed to all running coroutines, leaving approximately 20% of the free CPU time to the browser itself for rendering and event processing. The API assists the developer in writing coroutines by offering functions that help coroutines release CPU at the right moment and the API helps in saving or restoring the state of a coroutine. However, that still leaves some work to be done by the developer of a coroutine.

Scheduler details

The scheduler ensures that each coroutine receives time to execute without consuming all available CPU time, so that there is enough time left for the browser to render the Web page(s), the UI and to process user input and to execute other JavaScript code. The scheduler implements the above-described design in software. Therefore the scheduler contains a global execution context. Whenever a new coroutine is created, the current execution context is "remembered" with the coroutine, and whenever a coroutine is called, a new execution context is created and its parent execution context is set to the execution context "remembered" with the coroutine at create time. The scheduler itself supports automatic priority management. So coroutines consuming a lot of CPU time are degraded while those not consuming any CPU time are upgraded. This ensures that coroutines consuming only a small amount of CPU time (for example those handling user input processing) are not delayed by coroutines consuming large amounts of CPU time (for example rendering jobs). The major advantage of the coroutines can be seen on mobile devices which have much less CPU power than desktop PCs.

Introduction to coroutines

Not every function can become a coroutine and a coroutine is not a normal function. As already mentioned, coroutines are cooperative, therefore there is no need for mutexes. A coroutine appears as if it were executed directly when it is called. A coroutine has to call the method `shallyield()` from time to time to check if it is time to release the CPU. If the `shallyield()` returns `true`, the coroutine must call the method `yield()` and return to the caller. Calling `yield()` signals the scheduler that the function is not finished, but wants to give way to some other code. Some time later, the scheduler calls the coroutine again to allow it to continue its work.

Because of this, coroutines must remember their state and execution position between calls. A coroutine can store variables including information about the current point of execution with the *variable object*. Additionally, it is possible for one coroutine to call another coroutine and then wait for the called coroutine to finish (otherwise both coroutines would execute in parallel). Note that it is also possible that a coroutine may not end at all, so that it is executed in an endless loop, for example, rendering functions do not stop unless the [nokia.maps.map.Display](#) class to which they belong is destroyed; instead, they just go to sleep if there is nothing for them to do.

The possible states of the execution context of a coroutine are:

here

- **terminated** - the coroutine is finished
- **running** - the coroutine is currently running, but this execution context is not necessarily being executed; multiple execution contexts can be marked as "running", but in fact, they can only run one after another in succession (see [nokia.maps.util.Coroutine.current](#))
- **blocking** - the coroutine is waiting for something, either waiting for a coroutine it has called or just sleeping for a moment

Simple coroutine examples

The following example shows a very simple coroutine that increments a counter and another coroutine that updates a div-tag with the current value of the counter:

```
// A shortcut for easier access to the coroutine class:
//
var Coroutine = nokia.maps.util.Coroutine;

// Create a new coroutine with the name "counter". It receives two
// arguments, the variable object and the execution context.
//
var counterCo = Coroutine.create("counterCo", function (scope, context) {
    // The following check is important, because if the parent
    // context contains a variable "counter" a simple check for
    // scope.counter would return true (scope chain) and then
    // using the counter of the parent context might lead to
    // errors.
    //
    if (!scope.has("counter")) scope.counter = 0;

    // Loop until counter reaches 1 million.
    //
    while (scope.counter < 1*1000*1000) {
        // Increment the counter.
        //
        scope.counter++;

        // If we shall yield now, do that.
        //
        if (Coroutine.shallYield()) return Coroutine.yield();
    }
});

// Now let's run an instance of the coroutine.
//
var counterCoCtx = counterCo();

// Create another coroutine to be used as render coroutine. This method
// receives as the first argument a reference to the counter coroutine
// execution context it is to track and to the div-tag into which the state
// of the counter is to be rendered.
var renderCo = Coroutine.create("renderCo", function (scope, context) {
    while (true) {
```

here

```
// Output the current counter by accessing the variable
// object of the counter coroutine and reading the value.
//
scope.div.innerHTML = "Current count is
"+scope.counterCoCtx.scope.counter;

// If the counter context is terminated, we terminate too.
//
if (scope.counterCoCtx.status === Coroutine.TERMINATED) break;

// Otherwise put this coroutine to sleep for 10ms and then return.
return Coroutine.sleep(10);
};
},
// "counterCoCtx" is bound to the first parameter of the coroutine,
// the variable "div" to the second parameter.
//
"counterCoCtx", "div");

// Call the rendering coroutine for two different div-tags, but
// the same counter.
//
var renderAContext = renderCo(counterCoCtx,
document.getElementById("outputA"));
var renderBContext = renderCo(counterCoCtx,
document.getElementById("outputB"));
```

This was a very simple example of how to write coroutines. The whole concept becomes much more complex if one coroutine needs to invoke another coroutine or if it is necessary to wait until a coroutine has calculated a return value. As you can see, each coroutine receives the reference to the *variable object*. This does not mean that it is not possible to use local variables, but it should be taken into account that the state of the real local variables cannot be saved between two calls to a coroutine and therefore they must be used with caution.

Complex coroutines

The example below shows a much more complex coroutine, where one coroutine calls another and waits until the called coroutine is finished before using the return value from the called coroutine.

```
// A shortcut to access the static coroutine class more easily.
//
var Coroutine = nokia.maps.util.Coroutine;

// Create a function that adds all numbers from "from" to "to" and returns
// the sum (and yes, we are aware that there are better ways to do it).
//
var addFromToCo = Coroutine.create("addFromToCo", function (scope) {
// If the "sum" variable is not yet defined, initialize it with 0.
//
if (!scope.has("sum")) scope.sum = 0;

// Loop until we've added all values.
while (scope.from <= scope.to) {
```

here

```
// Add the value of "from" to our "sum".
//
scope.sum += scope.from++;

// If we are to release the CPU, do it.
if (Coroutine.shallYield()) return Coroutine.yield();
}

// If we're finished, return the calculated sum.
return scope.sum;
}, "from", "to");

// Create another coroutine to add a number from "a" to "b" "n" times.
//
var addMultiCo = Coroutine.create("addMultiCo", function (scope) {
  // In this case, everything gets a bit more complicated and we
  // have not only to remember some state variables, but also
  // the current execution point of the method (which we call "ip"
  // -- short for "instruction pointer") when the function is
  // called for the first time

  // When the method is called for the first time, the variable "sum"
  // contains the calculated sum, "i" contains the counter to count to
  // "n", and "addFromToCo" is a reference to an instance of the coroutine
  // that calculates the sum from "a" to "b".
  //
  if (!scope.has("ip")) {
    scope.sum = 0;
    scope.i = 0;
    scope.ip = 1;
  }

  // Now let's run a loop from "i" to "n".
  //
  while (scope.i < scope.n) {
    // If the current instruction pointer is 1, we have to call
    // the "addFromTo" coroutine now
    //
    if (scope.ip===1) {
      // and then the instruction pointer is 2.
      //
      scope.addFromToCtx = addFromToCo(scope.a, scope.b);
      scope.ip = 2;
    }

    // If the instruction pointer is currently 2
    //
    if (scope.ip===2) {
      // but the "addFromTo" coroutine is not yet finished
      //
      if (scope.addFromToCtx.status !== Coroutine.TERMINATED) {
        // We have to wait until the "addFromTo" coroutine
        // has finished processing. Therefore the "wait" call
        // lets us wait until the coroutine fires a
        // notify or broadcast to all blocking coroutines,
        // which automatically happens if the coroutine is
```

here

```
// terminated for whatever reason.
//
return Coroutine.wait(scope.addFromToCtx);
}

// If we've reached this point the coroutine "addFromTo" has
// been called and is finished, therefore we can read the returned
// value (which is a sum) and add it to the sum variable.
//
scope.sum += scope.addFromToCtx.returnValue;

// Now let's increase the loop counter and reset the instruction
// pointer to 1, so that the "addFromTo" coroutine is called
// again in the next loop iteration.
scope.i++;
scope.ip = 1;
}
}

// If we reach this point, the variable "i" has reached "n" and we're
// finished, so let's return the calculated sum.
//
return scope.sum;
}, "a", "b", "n");

// Let's call the "addMulti" coroutine to add the numbers from 1 to 1000
// and
// this 1000 times. Wait until the coroutine is finished by registering an
// observer for the "status" property of the execution context.
//
addMultiCo(1, 1000, 1000).addObserver(
    "returnValue",
    function (context, key, value, oldValue) {
        alert(context.returnValue);
    }
);
```

This example should make one thing clear: It is very easy to use coroutines for a normal user, that is at application level, but it is much harder to write them. The user just needs to call the coroutine like a normal function, except he is interested in the return value. To obtain it, he calls the method and registers an observer for the "returnValue" property, waiting until the method is finished so that he can then process the return value asynchronously.

Constructor Details

nokia.maps.util.Coroutine(name, fn)

This method is the constructor for the class.

It creates an instance of the class on the basis of the arguments supplied by the caller.

Parameters:

here

name:	{String}
	The name of the coroutine
fn:	{Function}
	The function that implements the coroutine

Property Details

static **AF_MAX_EXEC_TIME**: {Number}

This property holds the maximal execution time for those coroutine being executed within the animation frame request (see [nokia.maps.util.Coroutine.ExecutionContext#useAnimationFrame](#) and [nokia.maps.util.Coroutine#MAX_EXEC_TIME](#) for details).

static **AF_SLICE_SIZE**: {Number}

This property holds the maximal size of each time slice for a coroutine being executed within the animation frame request (see [nokia.maps.util.Coroutine.ExecutionContext#useAnimationFrame](#) and [nokia.maps.util.Coroutine#SLICE_SIZE](#) for details).

static constant **BLOCKED**: {Object}

This property holds a special object returned by the `sleep()`, `wait()` and `suspend()` methods.

static constant **BLOCKING**: {Number}

Coroutine status constant indicating that a context is blocked at a condition or sleeping.

static constant **KILLED**: {Object}

This property holds a special object returned by the `kill()` method.

static **MAX_EXEC_TIME**: {Number}

This property holds the maximum execution time or the number of milliseconds consumed by coroutines before control is handed over to the browser and so to other JavaScript code.

here

The smaller this time slice becomes, the more often the browser regains control and the smoother the execution of coroutines, so the browser appears to be more responsive, but as more overhead is produced and finally as less CPU time is used for the API, so as slower the API execution.

```
static constant RUNNING: {Number}
```

Coroutine status constant indicating that a context is currently running.

```
static SLICE_SIZE: {Number}
```

This property holds the maximum size of each time slice for a coroutine.

This value determines how many coroutines can run per scheduled time period, so it decides how long it takes until a coroutine gets a slice of CPU time. If this value is too big and a large number of coroutines run simultaneously, it can take seconds until each coroutine gets some CPU time, so tasking becomes unsteady, but performance is improved as fewer coroutine switches occur. If this value is too small and a large number of coroutines run at simultaneously, each coroutine gets a slice of CPU time faster, but the slice is also smaller, so execution of coroutines may seem to smoother, but the amount of CPU time lost due to coroutine switching increases, so the overall performance decreases.

This value is normally exactly 25% of the maximum execution time and you should keep it around that value. However, if you decrease the maximum execution time below 50ms you should consider increasing this value to 100%, so the same value as the maximum execution time and if you increase the maximum execution time to above 300ms you should consider changing this value to 10% of the maximum execution time (e.g. 75 for 300).

```
static constant TERMINATED: {Number}
```

Coroutine status constant indicating that a context is terminated.

```
static constant YIELDED: {Object}
```

This property holds a special object returned by the `yield()` method.

Method Details

```
static all(): {nokia.maps.util.Coroutine.ExecutionContext[]}
```

This method returns an array with all the execution contexts currently marked as "running" or "blocking".

here

Returns:

`{nokia.maps.util.Coroutine.ExecutionContext[][]}`

An array with all execution contexts in the state "running" or "blocking", the array may be empty if no context is currently executed

static **broadcast**(obj)

This method signals all coroutines blocking on an object and resumes them.

Parameters:

obj: {Object}

The object on which to resume all blocking coroutines.

Example:

```
var Coroutine = nokia.maps.util.Coroutine,
    condition = {},
    myCo = Coroutine.create("myCo", function (scope, context) {
        // If this is the first run ...
        if (!scope.ip) {
            // ... block on the condition.
            scope.ip = 1;

            return Coroutine.wait(condition);
        }
        alert("coroutine "+ context.name+" is now awake!");
        // coroutine terminates here.
    });

// Call the coroutine twice.
myCo();
myCo();

// After two seconds, wake all waiting coroutines.
setTimeout(function () {
    Coroutine.broadcast(condition);
}, 2000);
```

static **catchException**(enable)

This method enables exception catching in the scheduler to ignore errors in coroutines or it can be used to disable exception catching for debugging purposes. By default, exceptions are caught and written into the debug console.

here

Parameters:

enable: {Boolean}
true enables exception catching, false disables it

static **count**(): {Number}

This method retrieves the number of contexts not being terminated, thus the return value includes all running and blocked contexts.

Returns:

{Number} The number of contexts that are not terminated

static **create**(name, fn, args): {Function}

This method creates a new coroutine, using the name and implementation supplied by the caller.

Parameters:

name: {String}
The name of the coroutine

fn: {Function}
The function that implements the coroutine

args ...: {String} [optional]
An unlimited number of strings with the names of the arguments of this function

Returns:

{Function} The coroutine with the supplied function as internal run-function

See: [nokia.maps.util.Coroutine](https://api.nokia.maps.com/api/1.0/nokia.maps.util.Coroutine)

static **createEx**(name, fn, defaults, args): {Function}

here

This method creates a new coroutine using the name and implementation supplied by the caller. It creates coroutine by [nokia.maps.util.Coroutine.create](#) call and links defaults to the coroutine context.

Parameters:

name:	{String}
	The name of the coroutine
fn:	{Function}
	The function that implements the coroutine
defaults:	{Object} [optional]
	A hashmap with default values for all created contexts, currently the only options being supported are priority and useAnimationFrame
args ...:	{String} [optional]
	An unlimited amount of strings with the names of the arguments of this function

Returns:

{Function}	The coroutine with the supplied function as internal run-function

See: [nokia.maps.util.Coroutine](#)

static **current()**: {[nokia.maps.util.Coroutine.ExecutionContext](#)}

This method returns the currently executed context or null if no context is executed right now.

Returns:

{ nokia.maps.util.Coroutine.ExecutionContext }	The currently executed context or null if no context is executed

static **forId(id)**: {[nokia.maps.util.Coroutine.ExecutionContext](#)}

This method retrieves the execution context with the given id or undefined if no such context exists.

here

Parameters:

id: {String}

The unique execution context id for which to search

Returns:

{*nokia.maps.util.Coroutine.ExecutionContext*}

The execution context with the given unique id or undefined if no context with that id exists

static **forName**(name): {*nokia.maps.util.Coroutine.ExecutionContext*[]}}

This method searches through the list of all contexts currently in the "running" or "blocking" state and returns all those that are bound to the coroutine with the name supplied by the caller.

Parameters:

name: {RegExp | String}

The name of the coroutine to find or the regular expression to match against the coroutine name

Returns:

{*nokia.maps.util.Coroutine.ExecutionContext*[]}}

An array with all contexts where the matching name was found or an empty array if no coroutine matches

static **kill**(context, reason): {Object}

This method immediately terminates the currently executed coroutine or the supplied coroutine context.

Parameters:

context: {*nokia.maps.util.Coroutine.ExecutionContext*} [optional]

The coroutine context to terminate; if not provided, the currently executed coroutine context is terminated, if executed

here

reason: {Object} [optional]

Reason for termination, if any; if the first argument is omitted, the second argument becomes the first and must not be an instance of *nokia.maps.util.Coroutine.ExecutionContext*.

Returns:

{Object} If the context is killed, the method returns a *nokia.maps.util.Coroutine.KILL* object, otherwise undefined

static **killAll()**

This method destroys all execution contexts currently in the "running" or "blocking" state.

The method is intended for debugging purposes or to be used when the user has left a Web site and resources are to be freed.

static **resume(context)**

This method resumes the supplied executed execution context.

It unblocks the context from any condition on which it is currently blocked or wakes it up if it is sleeping. If it is a realtime context, the context becomes the next one to be executed, otherwise it is added at the end of the pending contexts.

Parameters:

context: {*nokia.maps.util.Coroutine.ExecutionContext*}

The execution context to resume

static **scheduler(isTestUnitCall)**

This method forces the scheduler to run at once.

Note that you should avoid calling this method, except if you have a very specific reason to do so, because by calling the scheduler directly, you bypass the normal scheduling and therefore you can cause the coroutine switching to become jerky and the browser might become unresponsive.

Parameters:

here

`isTestUnitCall:` `{Boolean}` [optional]

True if this is a test unit call, false otherwise (this parameter may only be used for internal test units).

`static schedulerCpuTime(): {Number}`

This method retrieves the total CPU time that the scheduler itself has consumed.

The returned value is not very precise, because some method calls are not be counted for the scheduler due to performance impact. If the context calls for example `yield()` or `sleep()`, this counts as context time, even though a small amount of that time normally should have be scheduler time.

Returns:

`{Number}` The number of milliseconds of CPU time the scheduler has consumed for itself

`static schedulerNextRuntime(): {Number}`

This method retrieves the UNIX timestamp showing when the scheduler should run again.

This together with the manual scheduler call can be used to work around browser bugs such as the bug in IE that causes it not to call any function registered using `setTimeout` or `setInterval` while the user drags with the left mouse button. To counteract this, for example, nokia.maps.dom.Page class implements a workaround that calls the scheduler within a `mousemove` or `drag` event if the current time is later then the scheduler's next run time.

Returns:

`{Number}` The UNIX timestamp showing when the scheduler should be executed again

`static scope(fn)`

This method clones the given function and moves the clone into the internal scope of the coroutine scheduler, removing the original scope remembered by the function.

This method is intended for debugging or hot-fixing purposes. It allows you to modify, read and replace internal methods and properties.

Parameters:

here

fn: {Function}

The function which is copied into the scope.

static **shallYield()**: {Boolean}

This method checks if the currently executed coroutine context must yield or not.

Returns:

{Boolean} true if the currently executed coroutine context is to yield control, otherwise false

static **signal(obj)**: {*nokia.maps.util.Coroutine.ExecutionContext*}

This method signals the given object and resumes the next coroutine that is blocked on this object, if any.

Parameters:

obj: {Object}

The object on which to signal the next blocking coroutine

Returns:

{*nokia.maps.util.Coroutine.ExecutionContext*}

Either the context that has been unblocked or undefined if no coroutine was blocking on this object

Example:

```
var Coroutine = nokia.maps.util.Coroutine,
    condition = {},
    myCo = Coroutine.create("myCo", function (scope, context) {
        // If this is the first run ...
        if (!scope.ip) {
            // ... block on the condition.
            scope.ip = 1;
            return Coroutine.wait(condition);
        }
        alert("coroutine is now awake!");
        // coroutine terminates here.
    });
// Call the coroutine.
```

here

```
myCo();
// In 2 seconds wake the coroutine up.
setTimeout(function () {
    Coroutine.signal(condition);
}, 2000);
```

static **sleep**(timeout, context): {Object}

This method puts a coroutine context to sleep.

Parameters:

timeout: {Number} [optional]

The number of milliseconds the coroutine context is to sleep; if omitted, the coroutine sleeps until resumed or terminated

context: {[nokia.maps.util.Coroutine.ExecutionContext](#)} [optional]

The execution context to put to sleep; if not given, the currently executed execution context is used, if any

Returns:

{Object} If the context is blocked, the method returns the [nokia.maps.util.Coroutine.BLOCKED](#) object, otherwise undefined

Example:

```
// Always use a "return" when calling Coroutine.sleep() to break
the current execution.
return Coroutine.sleep(10000, myCoCtx);
```

static **suspend**(context, timeout): {Object}

The method suspends the supplied execution context or the currently executed context.

Parameters:

context: {[nokia.maps.util.Coroutine.ExecutionContext](#)} [optional]

The execution context that to suspend; if omitted or null/undefined, the currently executed context is suspended, if any

here

timeout: {Number} [optional]

The maximum amount of time to block in milliseconds; if null or undefined, the method blocks until the coroutine is notified, resumed or terminated

Returns:

{Object} If the context is blocked, the method returns the [nokia.maps.util.Coroutine.BLOCKED](#) object, otherwise undefined

Example:

```
// Always use a "return" when calling Coroutine.suspend() to
break the current execution.
return Coroutine.suspend(myCoCtx);
```

static taskCpuTime(): {Number}

This method retrieves the total CPU time that all contexts have consumed.

Returns:

{Number} The number of milliseconds of CPU time all contexts have consumed together

static totalCpuTime(): {Number}

This method retrieves the number of milliseconds since the scheduler was started.

Returns:

{Number} The number of milliseconds since the scheduler was started

static wait(obj, timeout, context): {Object}

This method blocks the a coroutine on the object provided by the caller.

Note that the object gets a new property called "\$blocking" which is managed by the coroutine manager and must not be modified.

Parameters:

here

obj: {Object}

The object on which the coroutine is to be blocked

timeout: {Number} [optional]

If provided, this argument is the maximum length of time in milliseconds that the coroutine blocks, otherwise if the argument is null or undefined, the method blocks for ever

context: {*nokia.maps.util.Coroutine.ExecutionContext*} [optional]

The execution context to block; if the argument is omitted, the currently executed context is used, if any is currently executed

Returns:

{Object} If the context is blocked, the method returns the *nokia.maps.util.Coroutine.BLOCKED* object; otherwise undefined

Example:

```
var Coroutine = nokia.maps.util.Coroutine,
    condition = {},
    myCo = Coroutine.create("myCo", function (scope, context) {
        // If this is the first run ...
        if (!scope.ip) {
            // ... block on the condition.
            scope.ip = 1;
            // Using a "return" when calling Coroutine.yield() we
            break the current execution
            return Coroutine.wait(condition);
        }
        alert("coroutine is now awake!");
        // coroutine terminates here.
    });
// Call the coroutine.
myCo();
// Wake up the coroutine in two seconds.
setTimeout(function () {
    Coroutine.signal(condition);
}, 2000);
```

static **yield**(context): {Object}

This method makes the currently executed coroutine or the supplied one yield.

here

Parameters:

context: {[nokia.maps.util.Coroutine.ExecutionContext](#)}

The execution context of the coroutine to yield, if not given the currently executed context is suspended

Returns:

{Object} If the context is blocked the method returns the [nokia.maps.util.Coroutine.YIELDED](#) object, otherwise undefined

Example:

```
var YIELD_EVERY = 64,
    yieldCount = 0,
    Coroutine = nokia.maps.util.Coroutine,
    myCo = Coroutine.create("myCo", function (scope, context) {
        console.log("yieldCount" + yieldCount);
        while (++yieldCount < 1000) {
            // Break the execution of the while loop, using the yield
            // Using a "return" when calling Coroutine.yield() we
            // break the current execution.
            if (!(yieldCount % YIELD_EVERY)) return Coroutine.yield();
            // coroutine terminates here.
        }
    });
myCo();
```

Class: [ExecutionContext](#)

This class is a member of [nokia.maps.util.Coroutine](#).

Class Summary

This class represents the execution context of a coroutine.

[For full details, see [nokia.maps.util.Coroutine.ExecutionContext](#)]

Table 264: Property Summary

Properties
callTime: {Number} This property holds the UNIX timestamp showing when the execution context was called.

here

Properties

constant **coroutine**: {*nokia.maps.util.Coroutine*}

This property holds the coroutine to which the given execution context belongs (calling a coroutine always returns an execution context).

constant **id**: {String}

This property holds the unique id of the given execution context.

constant **name**: {String}

This property holds the name of the coroutine that is being executed for this execution context.

constant **parent**: {*nokia.maps.util.Coroutine.ExecutionContext*}

This property holds a reference to the parent execution context or null if this is the global execution context.

realtime: {Boolean}

If this property is set to true, it has an effect on the scheduler and on how a context is resumed.

returnValue: {Object}

This property hold the value returned by the coroutine of the execution context at termination.

constant **scope**: {*nokia.maps.util.Coroutine.Scope*}

This property holds an object that contains the variable values used within the coroutine.

status: {Number}

This property holds the status of the execution context.

timeStamp: {Number}

This property holds the latest UNIX timestamp known for the given execution context.

useAnimationFrame: {Boolean}

If this property is set to true, the scheduler moves the coroutine into the animation queue.

Table 265: Method Summary

Methods

nice () : {Number}

This method retrieves the current nice value for the given coroutine.

onError (scope, context, error) : {Exception}

If a context has a method onError() attached to it, the method is called as soon as an uncaught exception has been raised somewhere within the coroutine of the context.

onTerminated (scope, context)

If a context has a method onTerminated() attached to it, this method is called as soon as the coroutine has finished.

run (synchron, force) : {Number}

here

Methods

The `run()` method of an execution context can be called to force immediate execution of the coroutine associated with the context.

Class Description

This class represents the execution context of a coroutine. The execution context of a coroutine is an emulated execution context, similar to the native execution context of a normal JavaScript function, with the difference that the coroutine execution context can be suspended and recovered. Therefore multiple JavaScript functions (coroutines) can run in parallel, cooperatively sharing the CPU. The execution context contains all the information about the state of an asynchronous function call (so about a coroutine call). The state information includes the coroutine status (running, blocking or terminated), the `returnValue` and the local variable stack (scope).

The execution context of a coroutine can be compared with the handle returned by the method `setInterval()`, but the execution context contains more information. The execution context allows data transfer between calls to the coroutine and permits all asynchronous coroutines to share CPU time cooperatively between them, keeping the browser responsive, while one or multiple complex operations are ongoing.

For a more detailed description of what an execution context is and how it behaves, please refer to the documentation for the class [nokia.maps.util.Coroutine](#).

Constructor Details

`nokia.maps.util.Coroutine.ExecutionContext(coroutine, args)`

Creates execution context.

Parameters:

`coroutine:` {[nokia.maps.util.Coroutine](#)}

`args:` {Object}

Property Details

`callTime:` {Number}

This property holds the UNIX timestamp showing when the execution context was called. The timestamp is valid only as long as the given execution context is the currently executing context.

constant **`coroutine:`** {[nokia.maps.util.Coroutine](#)}

here

This property holds the coroutine to which the given execution context belongs (calling a coroutine always returns an execution context).

constant id: {String}

This property holds the unique id of the given execution context.

constant name: {String}

This property holds the name of the coroutine that is being executed for this execution context.

constant parent: {*nokia.maps.util.Coroutine.ExecutionContext*}

This property holds a reference to the parent execution context or `null` if this is the global execution context.

realtime: {Boolean}

If this property is set to `true`, it has an effect on the scheduler and on how a context is resumed.

Normally the scheduler sleeps for at least a specific amount of milliseconds before it continues to execute coroutines. The reason is that the browser needs this time for rendering and event processing. If a coroutine context is set to `realtime`, the scheduler ensures that it is executed exactly (as far as possible) at the desired time, regardless of the risk that the browser might receive only a few milliseconds or no time at all for rendering and event processing. Furthermore, a real time context is put at the front of the round robin list and always ahead of any non-realtime contexts in the priority list.

This means that realtime contexts may make the browser unresponsive and slow down all other normal coroutine contexts. Therefore, we highly recommend that you do not make contexts realtime unless there is a very good reason to do so.

returnValue: {Object}

This property hold the value returned by the coroutine of the execution context at termination.

constant scope: {*nokia.maps.util.Coroutine.Scope*}

here

This property holds an object that contains the variable values used within the coroutine. For example, the object includes the property "arguments".

status: {Number}

This property holds the status of the execution context. The possible values are *nokia.maps.util.Coroutine.TERMINATED*, *nokia.maps.util.Coroutine.RUNNING* or *nokia.maps.util.Coroutine.BLOCKING* (default).

timeStamp: {Number}

This property holds the latest UNIX timestamp known for the given execution context. The property is updated whenever the execution of the execution context begins and whenever the context calls the method `shallYield()`.

If the `shallYield()` is regularly called by a coroutine, then this property always contains the current UNIX timestamp.

useAnimationFrame: {Boolean}

If this property is set to `true`, the scheduler moves the coroutine into the animation queue. This means that the coroutine is only executed within the `requestAnimationFrame` callback. See also W3C <http://www.w3.org/TR/animation-timing/>. We strongly recommend setting this property to `true` if a method has a large CPU impact.

The animation queue does not follow the normal scheduling process, because it runs in parallel to the normal process and has only a priority list. If multiple rendering tasks run at the same time, the most important one may be done first, while the others have to wait until it has finished. However, because the execution of a coroutine reflects its priority, an exact execution order cannot be guaranteed. It can be said that the coroutine that is executed the fastest and has the highest priority is executed first, while the coroutine with the lowest priority that consumes the most time is executed last.

Another difference between the scheduling of the animation queue and normal scheduling is that sleep or wait times are not precise, because the rendering is bound to the `requestAnimationFrame` method of the browser. If a browser tab is in the background, the browser reduces the number of calls to the `requestAnimationFrame()` method dramatically and may even cut it down to zero, therefore reducing the amount of time used for rendering. This helps save battery power on mobile devices, because while a user is not viewing a page no animation frame requests are performed. Therefore even methods that have nothing to do with rendering,

here

but are very expensive (for example heavy calculations) may be marked as `useAnimationFrame` to decrease CPU usage on mobile devices and to save battery power.

Note that if this property is set, `precede` and `postpone` have no effect for the given coroutine context.

Method Details

`nice()`: {Number}

This method retrieves the current nice value for the given coroutine.

The nice value is the final absolute priority with which the scheduler handles the context. When the nice value is higher, the context acquires CPU time faster. The nice value is limited between -200 (bad context consuming a lot of CPU time with a very low priority) and +100 (good context, not consuming much CPU time, but has a very high priority).

The scheduler executes one context from the round robin list and then the context with the highest nice value from the priority list. Afterwards, it decrements the nice value and increments the activity value.

This has been designed to ensure that each context receives at least a small amount of CPU time (50% is given away in equal chunks to all contexts, regardless of priority) while contexts with a higher priority and those that are often idle receive the other 50% of CPU time, depending on their priority and activity (high priority and low activity comes first).

Activity is taken into account for the reason that there may be contexts waiting for user input. Those contexts have low activity levels, but are very important with regard to user satisfaction, because the user expects a fast reaction to his input. Therefore, activity is a very important point for the scheduler. If a context has a very high priority but it is consuming all the CPU time given to it, it receives a lower priority than an idle context with a very low priority (unless the context with the lower priority begins to consume CPU time as well).

The nice value is calculated simply as priority minus activity, limiting it to a value between -200 (bad) and +100 (good).

Note that the nice value returned by this method is not be updated in real-time for performance reasons. While a coroutine is idle, the nice value decreases, but this is not reflected by this method. Therefore the value returned by the method might not be accurate.

Returns:

{Number} The current nice value of this coroutine (not accurate)

here

onError(scope, context, error): {Exception}

If a context has a method `onError()` attached to it, the method is called as soon as an uncaught exception has been raised somewhere within the coroutine of the context. The method sets the reason for the exception and returns `null` or `undefined`, or it returns the exception itself to signal that it was not able to determine the reason for the error and that the context is being terminated.

Parameters:

scope:	{Object}
	The variable object representing the scope from the context
context:	{ <i>nokia.maps.util.Coroutine.ExecutionContext</i> }
	The execution context
error:	{Exception}
	The exception that was the reason for the error

Returns:

{Exception}	The exception object if the method cannot determine the reason for the exception, <code>null</code> or <code>undefined</code> to indicate that the reason for the exception has been determined and/or that the exception should be ignored by the scheduler

onTerminated(scope, context)

If a context has a method `onTerminated()` attached to it, this method is called as soon as the coroutine has finished.

Parameters:

scope:	{Object}
	The variable object representing the scope from the context
context:	{ <i>nokia.maps.util.Coroutine.ExecutionContext</i> }
	The execution context

here

run(synchron, force): {Number}

The `run()` method of an execution context can be called to force immediate execution of the coroutine associated with the context. The method is used internally by the scheduler to run a context. It should not be called by third-party applications, except, where there is an important reason to run the context immediately and synchronously.

The `run()` method removes the coroutine context from all internal lists before the code of the coroutine is executed.

Parameters:

synchron: {Boolean} [optional]

If this argument is provided and `true`, the method `shallyield()` never returns `true`, therefore the context is executed synchronously; please be aware that there is no guarantee that the coroutine will not put itself to sleep unless it supports synchronous calls

force: {Boolean} [optional]

If this argument is `true` the method will ignore the precedence and postpone settings as well as the `useAnimationFrame` option, otherwise running the method will not do anything as long as it is postponed to any other method or we're out of an animation frame.

Returns:

{Number} A UNIX timestamp indicating when the method finished executing

Class: Scope

This class is a member of [nokia.maps.util.Coroutine](#).

Class Summary

Scope is an abstract class designed to hold variables available within the scope of a coroutine.

[For full details, see [nokia.maps.util.Coroutine.Scope](#)]

Table 266: Property Summary

Properties
arguments : {Object[]}

here

Properties

An array with all the parameters that have been passed to the coroutine.

context: {*nokia.maps.util.Coroutine.ExecutionContext*}

The execution context that is bound to the given scope.

global: {*nokia.maps.util.Coroutine.Scope*}

The reference to the scope of the global execution context.

that: {Object}

The object to which this pointed at the time the coroutine was called.

Table 267: Method Summary

Methods

hasOwn (name) : {Boolean}

This method checks if the variable named by the caller is defined in the given scope.

set (name, value) : {*nokia.maps.util.Coroutine.ExecutionContext*}

This method sets the value of a variable on the basis of the variable name and value supplied by the caller.

Class Description

Scope is an abstract class designed to hold variables available within the scope of a coroutine.

Property Details

arguments: {Object[]}

An array with all the parameters that have been passed to the coroutine.

context: {*nokia.maps.util.Coroutine.ExecutionContext*}

The execution context that is bound to the given scope.

global: {*nokia.maps.util.Coroutine.Scope*}

The reference to the scope of the global execution context.

that: {Object}

The object to which this pointed at the time the coroutine was called.

here

Method Details

hasOwn(name): {Boolean}

This method checks if the variable named by the caller is defined in the given scope. The method is an alias for the W3C method `hasOwnProperty()`. It returns a Boolean indicating whether a variable with the given name is defined at this scope. The method is an alias to W3C's `hasOwnProperty()`.

Parameters:

name: {String}

The string containing the name of the property to be tested.

Returns:

{Boolean} true if the property exists in this scope, otherwise false

set(name, value): {*nokia.maps.util.Coroutine.ExecutionContext*}

This method sets the value of a variable on the basis of the variable name and value supplied by the caller. The method searches all scopes in the scope chain and sets the the value of the first variable with a matching name. If no matching variable exists, the method creates it within the global scope.

This method simulates the behavior of the normal assignment in JavaScript, example:

```
var Coroutine = nokia.maps.util.Coroutine,
    outerFn = function () {
        var text = "Old Value",
            innerFn = function () {
                // Modifies the "text" variable from the parent scope:
                text = "New Value";
            };
        innerFn();
        alert(text);
    };
var outerCo = Coroutine.create("outer", function (scope) {
    scope.text = "Old Value";
    scope.innerCo = Coroutine.create("inner", function (scope) {
        // The assignment
        //   scope.text = "New Value";
        // results in a new variable with the value "New Value"
        // in the current scope and does not change the variable
        // named 'text' in the parent scope. Therefore to change
        // the value of the variable 'text' in the parent scope,
        // it is necessary to use the method set():
        scope.set("text", "New Value");
    });
});
```

here

```
scope.ctx = scope.innerCo();
scope.ctx.run();
alert(scope.text);
});
outerFn();
outerCo();
```

Parameters:

name: {String}

The string containing the name of the property to be set

value: {Object}

The value to be assigned to property specified by the argument name

Returns:

{[nokia.maps.util.Coroutine.ExecutionContext](#)}

A reference to the given execution context for chaining (this)

Class: Event

This class is a member of [nokia.maps.util](#).

Class Summary

Simplified Event class without bubbling.

[For full details, see [nokia.maps.util.Event](#)]

Table 268: Property Summary

Properties	
readonly defaultPrevented : {Boolean}	This property indicates whether <code>preventDefault()</code> has been called for this event.
readonly target : { nokia.maps.util.EventTarget }	This property holds the target object of the event.
timeStamp : {Number}	This property specifies the time at which the event was created in milliseconds relative to 1970-01-01T00:00:00Z.
readonly type : {String}	This property holds the local name of the event type.

here

Table 269: Method Summary

Methods
<i>preventDefault</i> () This method cancels the default action for the given event.

Class Description

This class implements a Event like the W3C interface but simplified (no bubble and capture phases) [Event](#) but with less complexity. See also [Concept event](#)

For more details about event dispatch please refer to the documentation of the class [nokia.maps.util.EventTarget](#).

Constructor Details

nokia.maps.util.Event(type)

Parameters:

type: {String}
The type of the Event

Property Details

readonly **defaultPrevented**: {Boolean}

This property indicates whether `preventDefault()` has been called for this event.

readonly **target**: {[nokia.maps.util.EventTarget](#)}

This property holds the target object of the event.

timeStamp: {Number}

This property specifies the time at which the event was created in milliseconds relative to 1970-01-01T00:00:00Z.

readonly **type**: {String}

This property holds the local name of the event type.

here

Method Details

preventDefault()

This method cancels the default action for the given event.

Class: EventTarget

This class is a member of [nokia.maps.util](#).

Class Summary

This class implements the W3C DOM Level 3 interface [EventTarget](#)

[For full details, see [nokia.maps.util.EventTarget](#)]

Table 270: Method Summary

Methods
addListener (type, callback) This method registers an event listener on the EventTarget
dispatch (evt) : {Boolean} This method dispatches an event.
removeListener (type, callback) This method removes an event listener.

Class Description

This class implements the W3C DOM Level 3 interface [EventTarget](#) It can be used as a mixin for any JavaScript class or it can be applied at runtime to any JavaScript object by simply patch the object to EventTarget. Note that for the sake of compatibility the W3C, some methods have been renamed. For example, `addEventListener()` becomes `addListener()`.

Usage example:

```
myObject = new nokia.maps.util.EventTarget()
myObject.addListener("customEvent", function (evt) {
    alert( evt.type );
});
// Now you can fire an event at "myObject":
myObject.dispatch(
    new nokia.maps.util.Event("customEvent")
);
```

The listener queue

here

Constructor Details

nokia.maps.util.EventTarget(obj)

The class constructor mixes the functionality of `EventTarget` into the argument object.

Parameters:

obj: {Object}
The object to be extended with `EventTarget` functionality

Method Details

addListener(type, callback)

This method registers an event listener on the `EventTarget`

Parameters:

type: {String}
Specifies the event type for which the listener is to be registered.

callback: {Function}
The function used as the callback for the event. This function is called whenever an event occurs of the event type for which the listener was registered. The function receives an event argument that contains contextual information about the event (see [nokia.maps.util.Event](#)).

dispatch(evt): {Boolean}

This method dispatches an event.

Parameters:

evt: {[nokia.maps.util.Event](#)}
An object representing the event to be dispatched

Returns:

here

`{Boolean}` Indicates whether any of the listeners which handled the event called `Event.preventDefault()`. If `Event.preventDefault()` was called, the return value is `false`, otherwise it is `true`

removeListener(type, callback)

This method removes an event listener. Calling `removeListener` with arguments that do not identify any currently registered listener on the `EventTarget` has no effect.

Parameters:

type: `{String}`

Specifies the event type associated with the event for which the listener was registered.

callback: `{Function}`

The callback function to be removed.

Interface: IArrows

This interface is a member of [nokia.maps.util](#).

Interface Summary

This interface encapsulates properties of arrows that can be drawn, for example, along a route polyline to indicate the direction of travel.

[For full details, see [nokia.maps.util.IArrows](#)]

Table 271: Property Summary

Properties
readonly color : <code>{String}</code> This property holds the value of the fill color used for the arrow shapes.
readonly frequency : <code>{Number}</code> This property indicates the frequency of arrow shapes.
readonly length : <code>{Number}</code> This property indicates the length of the arrow shapes.
readonly width : <code>{Number}</code>

here

Properties

This property indicates the width of the arrow shape.

Interface Description

This interface represents the common properties of arrow objects that can be drawn on the map, typically superimposed on a polyline representing a route to indicate the direction of travel. An implementation of this interface can be used in [nokia.maps.map.Polyline](#) to define the arrows style.

Property Details

readonly **color**: {String}

This property holds the value of the fill color used for the arrow shapes. The format is RGBA (the alpha value is optional). When defining the RGBA value, please follow the CSS specification, which requires a string containing:

- a leading "#" character (followed by ...)
- two or one hex digits for every channel

Default Value: "#FFFC"

readonly **frequency**: {Number}

This property indicates the frequency of arrow shapes. The value is taken as factor of the length of the arrow. The value must be greater than 0 (zero). A value of 1 results in arrows without spaces between them.

Default Value: 5

readonly **length**: {Number}

This property indicates the length of the arrow shapes. The value is taken as a factor of the width of the line at the end of which the arrow is drawn. The value must be greater than 0.

Default Value: 1.6

readonly **width**: {Number}

here

This property indicates the width of the arrow shape. The value is taken as a factor of the width of the line, where the arrow description is applied. The value must be greater than 0.

Default Value: 1.2

Interface: IBox

This interface is a member of [nokia.maps.util](#).

Interface Summary

This interface defines a box.

[For full details, see [nokia.maps.util.IBox](#)]

Table 272: Property Summary

Properties
bottom : {Number} This property holds the numeric value for the bottom side of the box.
left : {Number} This property holds the numeric value for the left side of the box.
right : {Number} This property holds the numeric value for the right side of the box.
top : {Number} This property holds the numeric value for the top side of the box.

Interface Description

This interface class defines a box with properties that hold the screen coordinates of the sides of the box: top, right, bottom and left.

Property Details

bottom: {Number}
This property holds the numeric value for the bottom side of the box.

left: {Number}
This property holds the numeric value for the left side of the box.

here

right: {Number}

This property holds the numeric value for the right side of the box.

top: {Number}

This property holds the numeric value for the top side of the box.

Interface: IBrush

This interface is a member of [nokia.maps.util](#).

Interface Summary

This interface represents brush objects.

[For full details, see [nokia.maps.util.IBrush](#)]

Table 273: Property Summary

Properties
readonly color: {String} This property holds the value of the fill color used for the brush.
readonly fill: {String} This property indicates the fill type to be used for the brush.

Interface Description

This interface represents the common properties of brush objects. An implementation of this interface can be used in [nokia.maps.map.Object](#) to define fill colors.

Property Details

readonly **color:** {String}

This property holds the value of the fill color used for the brush. The format is RGBA (the alpha value is optional). When defining the RGBA value, please follow the CSS specification, which requires a string containing:

- a leading "#" character (followed by ...)
- two or one hex digits for every channel

here

Default Value: "#00F8"

readonly **fill**: {String}

This property indicates the fill type to be used for the brush. The value should be one of the following:

- "solid" - indicates a solid fill
- "none" - indicates no fill

Default Value: "solid"

Interface: IPen

This interface is a member of [nokia.maps.util](#).

Interface Summary

This interface class defines a IPen object.

[For full details, see [nokia.maps.util.IPen](#)]

Table 274: Property Summary

Properties
readonly lineCap : {String} This property specifies the shape to be used at the end of open subpaths when they are stroked.
readonly lineJoin : {String} This property specifies the shape to be used on the corners of paths or basic shapes when they are stroked.
readonly lineWidth : {Number} This property holds the width of lines in coordinate space units.
readonly miterLimit : {Number} The miter length is the distance from the point, where the join occurs to the intersection of the line edges on the outside of the join.
readonly stroke : {String} This property specifies the lines around shapes.
readonly strokeColor : {String} This property represents the color to use for the lines drawn around shapes.

here

Interface Description

This interface class defines a IPen object. IPen object is used in [nokia.maps.map.Object](#) to define outline color.

Property Details

readonly **lineCap**: {String}

This property specifies the shape to be used at the end of open subpaths when they are stroked. The value should be one of the following:

- "butt"
- "round"
- "square"

Default Value: "round"

readonly **lineJoin**: {String}

This property specifies the shape to be used on the corners of paths or basic shapes when they are stroked. The value should be one of the following:

- "miter"
- "bevel"
- "round"

Default Value: "miter"

readonly **lineWidth**: {Number}

This property holds the width of lines in coordinate space units.

Default Value: 1

readonly **miterLimit**: {Number}

The miter length is the distance from the point, where the join occurs to the intersection of the line edges on the outside of the join. The miter limit ratio is the maximum allowed ratio of the miter length to half the line width. If the miter length were to cause the miter limit ratio to be exceeded, this second triangle must not be rendered.

here

Default Value: 10

readonly **stroke**: {String}

This property specifies the lines around shapes. The value should be one of the following:

- "solid"

Default Value: "solid"

readonly **strokeColor**: {String}

This property represents the color to use for the lines drawn around shapes.

Default Value: "#00F8"

Interface: IPoint

This interface is a member of [nokia.maps.util](#).

Interface Summary

This interface defines the property of a point object in a two-dimensional coordinate space.

[For full details, see [nokia.maps.util.IPoint](#)]

Table 275: Property Summary

Properties
readonly x : {Number} The x coordinate of the point.
readonly y : {Number} The y coordinate of the point.

Interface Description

This interface defines the property of a point object in a two-dimensional coordinate space.

Property Details

readonly **x**: {Number}

here

The x coordinate of the point.

readonly **y**: {Number}

The y coordinate of the point.

Class: OList

This class is a member of [nokia.maps.util](#).

Class Summary

This class defines an observable list.

[For full details, see [nokia.maps.util.OList](#)]

Table 276: Method Summary

Methods
add (element, idx) : { nokia.maps.util.OList } This method adds a new element to the list.
addAll (elements, idx) : { nokia.maps.util.OList } This method adds multiple new elements to the list.
addObserver (callback, context) This method registers a new observer on the given instance of OList.
asArray () : {Array} This method gets the list as an array.
clear () This method clears all elements from the list.
get (idx) : {Variant} This method gets the value for the specified list element.
getLength () : {Number} This method getg the length of the list.
indexOf (obj) : {Number} This method retrieves the index of the first object in this list that matches the object supplied by the caller.
remove (obj) : {Variant} This method removes from the list the first element that matches the object supplied by the caller.
removeAll (elements) : { nokia.maps.util.OList }

here

Methods

This method removes multiple elements from the list.

`removeAt` (idx) : {Variant}

This method removes an element by a given index from the list.

`removeObserver` (callback, context) : {*nokia.maps.util.OObject*}

This method unregisters an observer.

`set` (element, idx) : {Variant}

This method sets (replaces) an element at the given index in the list.

Class Description

This class defines an observable list.

Constructor Details

`nokia.maps.util.OList`(elements)

This method initializes a new instance of the class.

Parameters:

elements: {Array} [optional, default: []]

The initial list elements to use; if omitted, an empty array is used

Method Details

`add`(element, idx): {*nokia.maps.util.OList*}

This method adds a new element to the list.

Parameters:

element: {Variant}

The new element to add

idx: {Number} [optional]

The index where the new element should be inserted; if omitted or greater than the current size of the list, the element is added at the end of the list; a negative index is treated as being relative from the end of the list

here

Returns:

`{nokia.maps.util.OLList}`

The given instance of `OLList`

Throws:

`{IllegalState}` When an observer triggers a rollback during an ongoing rollback

`{IllegaleArgument}` When the argument `idx` is not a number

addAll(elements, idx): `{nokia.maps.util.OLList}`

This method adds multiple new elements to the list.

Parameters:

elements: `{Variant[]}`

The new elements to add

idx: `{Number}` [optional]

The index where the new elements should be inserted; if omitted or greater than the current size of the list, the elements are added at the end of the list; a negative index is treated as being relative from the end of the list

Returns:

`{nokia.maps.util.OLList}`

The given instance of `OLList`

Throws:

`IllegalState` if an observer triggers a rollback during an ongoing rollback

addObserver(callback, context)

This method registers a new observer on the given instance of `OLList`. The observer is a callback to be invoked when the instance changes, typically when an element is added or removed.

here

Parameters:

callback: {Function}

The callback function to be invoked after the list has been modified; the callback must accept the following arguments:

- (OList) oList - the OList instance itself
- (String) operation - the name of the operation which triggered a change to the given instance of OList and therefore a call to the callback. The operation is either "add" or "remove"
- (Variant) element - the element which was added or deleted
- (Number) idx - the index of the element concerned

The callback can signal that a rollback should be triggered by returning true.

context: {Object} [optional]

The context to use when the callback is invoked. If not set than global context is used.

asArray(): {Array}

This method gets the list as an array.

Returns:

{Array} The list as an array

clear()

This method clears all elements from the list.

Throws:

IllegalState if an observer triggers a rollback during an ongoing rollback

get(idx): {Variant}

This method gets the value for the specified list element.

here

Parameters:

idx: {Number}
The index of the element to get

Returns:

{Variant} The value of the element

Throws:

IllegaleArgument if an observer triggers a rollback during an ongoing roll-back

getLength(): {Number}

This method getg the length of the list.

Returns:

{Number} The length of the list

indexOf(obj): {Number}

This method retrieves the index of the first object in this list that matches the object supplied by the caller.

Parameters:

obj: {Variant}
The object for which to return the index.

Returns:

{Number} The index of the first matching object in this list or -1 if the object provided by the caller is not found in the list

remove(obj): {Variant}

This method removes from the list the first element that matches the object supplied by the caller.

here

Parameters:

obj: {Variant}
The element to remove

Returns:

{Variant} The removed element

Throws:

IllegalState if an observer triggers a rollback during an ongoing rollback.

removeAll(elements): {*nokia.maps.util.OList*}

This method removes multiple elements from the list.

Parameters:

elements: {Variant[]}
The elements to remove

Returns:

{*nokia.maps.util.OList*}
The given instance of OList

Throws:

IllegalState if an observer triggers a rollback during an ongoing rollback

removeAt(idx): {Variant}

This method removes an element by a given index from the list.

Parameters:

idx: {Number}
The index of the element which should be removed

here

Returns:

{Variant} The removed element or null if a observer signaled a rollback

Throws:

IllegalState if an observer triggers a rollback during an ongoing rollback

removeObserver(callback, context): {*nokia.maps.util.Object*}

This method unregisters an observer.

Parameters:

callback: {Function}

The observer method to be removed

context: {Object}

The context in which the given callback should be called

Returns:

{*nokia.maps.util.Object*}

The reference to the given Object instance

set(element, idx): {Variant}

This method sets (replaces) an element at the given index in the list.

Parameters:

element: {Variant}

The element to set at the index specified by idx

idx: {Number}

The index of the element which should be replaced

Returns:

{Variant} The replaced element or undefined if an observer has triggered a rollback

here

Throws:

IllegalState if an observer triggers a rollback during an ongoing rollback

Class: OObject

This class is a member of *nokia.maps.util*.

Class Summary

This class encapsulates an inheritable object that has observable properties.

[For full details, see *nokia.maps.util.OObject*]

Table 277: Method Summary

Methods
<i>addObserver</i> (key, callback, context) : { <i>nokia.maps.util.OObject</i> } This method registers an observer for the property named by the caller.
<i>get</i> (key) : {Variant} This method retrieves the value of the property with the name provided by the caller.
<i>remove</i> (key) : { <i>nokia.maps.util.OObject</i> } This method removes the property with the name provided by the caller.
<i>removeObserver</i> (key, callback, context) : { <i>nokia.maps.util.OObject</i> } This method removes the observer for the property named by the caller.
<i>set</i> (nameOrObject, value, force) : { <i>nokia.maps.util.OObject</i> } This method sets property values, using the property names and values supplied by the caller.

Class Description

This class encapsulates an inheritable object that has observable properties.

Constructor Details

nokia.maps.util.OObject(props)

This method initializes an instance of *OObject* that is inheritable and has observable properties.

Parameters:

props: {Object} [optional]

here

The initial hash of properties with names and values.

Method Details

addObserver(key, callback, context): {*nokia.maps.util.OOobject*}

This method registers an observer for the property named by the caller. Note that modifying the property from within the observer is allowed, but discouraged, because this forces the observer chain to be invoked again and the old one aborted. This could lead to endless recursions and should be done carefully.

Parameters:

key: {String}

The name of the property to observe; the wild card "*" can be used to observe any property

callback: {Function}

The function to be called if the observed property is modified; the function must be able to receive the following arguments:

- (OOobject) obj - a reference to the given object
- (String) key - the name of the property that was modified, created or deleted
- (Variant) value - the new value that the property should be set to
- (Variant) oldValue the old value of the property

context: {Object} [optional, default: null]

The context in which the given function should be called. If omitted, the global object is used.

Returns:

{*nokia.maps.util.OOobject*}

A reference to the given OOobject instance

get(key): {Variant}

This method retrieves the value of the property with the name provided by the caller.

here

Parameters:

key: {String}
The name of the property whose value is to be retrieved

Returns:

{Variant} The retrieved value of the property named by the caller or undefined if no such property exists

remove(key): {*nokia.maps.util.Object*}

This method removes the property with the name provided by the caller.

Parameters:

key: {String}
The name of the property to be removed

Returns:

{*nokia.maps.util.Object*}
The reference to the given Object instance

removeObserver(key, callback, context): {*nokia.maps.util.Object*}

This method removes the observer for the property named by the caller.

Parameters:

key: {String}
The name of the property that should no longer be observed

callback: {Function}
The observer method to be removed

context: {Object} [optional]
The context in which the given callback should be called

here

Returns:

`{nokia.maps.util.Object}`

The reference to the given instance of *Object*

set(nameOrObject, value, force): `{nokia.maps.util.Object}`

This method sets property values, using the property names and values supplied by the caller. It takes an object whose keys match property names. The values corresponding to the keys are the new values for the properties to set.

To set a single property, the caller can provide two arguments, the name of the property as a string and the value to be set. For all properties, the following rules apply:

- If no property with the given name exists yet, the property is created.
- If a derived class defines a member method with the name of the key postfixed by "Setter", this method is called first; it is given the value and should return the "filtered" value to be set.
- After this "filtering" all registered observers of this value are called - if and only if this filtered value differs from the old value. The "observers" MUST NOT use code that leads to secondary setting of properties in this object/property or elsewhere, as this could result in endless loops.

The members of the object passed to `set()` are not tested against `hasOwnProperty()`.

Parameters:

nameOrObject: `{String | Object}`

The name of the property to be set or the object from which to take the properties and their values

value: `{Variant} [optional]`

The value that should be applied to the property; if this argument is present, nameOrObject is assumed to hold the name of a property

force: `{Boolean} [optional]`

A flag to force an assignment, even if the old value is identical to the new value (default false); the flag is taken into account only on setting a single property via name, value

Returns:

here

`{nokia.maps.util.OOobject}`

A reference to the given OOobject instance

Example:

```
// Setting a single property:
obj.set("foo", 123);

// Setting multiple properties at once:
obj.set({ foo: 123, bar: "second one" });

// Definition of a "setter" within an object
// that has OOobject in its prototype chain:
{
    ...
    fooSetter: function (value) {
        this.wantsToBeTriggered(); // if "foo" changes

        // ensure the value stored to member "foo" is pure int
        return value >>> 0;
    },
    ...
}
```

Class: Pen

This class is a member of [nokia.maps.util](#).

Class Summary

This class defines properties of pen objects that can be used when drawing shape and text.

[For full details, see [nokia.maps.util.Pen](#)]

Property Summary

Directly Inherited Properties

Inherited from class [nokia.maps.util.IPen](#) :

[lineCap](#), [lineJoin](#), [lineWidth](#), [miterLimit](#), [stroke](#), [strokeColor](#)

Class Description

This class defines properties of pen objects that can be used when drawing shape and text.

here

Constructor Details

nokia.maps.util.Pen(props, pen)

This method initializes a Pen object, using default values if property values are not specified by the caller.

Parameters:

- props:

{[nokia.maps.util.IPen](#)}

An object that specifies the initial values for the properties it names; default values are used for any properties that are not named in this object
- pen:

{[nokia.maps.util.Pen](#)} [optional]

An object from which to create a new instance of Pen

Class: Point

This class is a member of [nokia.maps.util](#).

Class Summary

This class represents a location within a two-dimensional space.

[For full details, see [nokia.maps.util.Point](#)]

Table 278: Property Summary

Properties
static isCoveredBy : {String Boolean} This method checks if the x/y coordinate is covered by the given shape.

Directly Inherited Properties

Inherited from class [nokia.maps.util.IPoint](#) :

[x](#), [y](#)

Table 279: Method Summary

Methods
add (other) : { nokia.maps.util.Point } This method obtains the sum of the given point and the point supplied by the caller.
ceil () : { nokia.maps.util.Point }

here

Methods

This method creates a new point whose x and y coordinates are obtained by rounding up the coordinates of the given point.

clone () : {*nokia.maps.util.Point*}

This method creates a exact clone of the given point.

distance (other) : {Number}

This method gets the distance to another point.

divide (divisor) : {*nokia.maps.util.Point*}

This method creates a new point whose x and y coordinates are obtained by dividing the coordinates of the given points by given divisor.

equals (other) : {Boolean}

This method checks if the given point is equal to the point supplied by the caller.

floor () : {*nokia.maps.util.Point*}

This method creates a new point whose x and y coordinates are obtained by rounding down the coordinates of the given point.

getNearest (first, second) : {*nokia.maps.util.Point*}

This method calculates a point on the line defined by the two points provided by the caller such that calculated point lies nearest to the given point.

isCoveredBy (path, lineWidth, closed) : {String | Boolean}

This method checks if the point is covered by the given polygon.

len () : {Number}

This method calculates the distance between the origin and the origin (x: 0, y: 0).

max (other) : {*nokia.maps.util.Point*}

This method creates a new point whose x and y coordinates are the largest values selected from the given point and the point supplied by the caller.

min (other) : {*nokia.maps.util.Point*}

This method creates a new point whose x and y coordinates are the smallest values selected from the given point and the point supplied by the caller.

modulo (divisor) : {*nokia.maps.util.Point*}

This method creates a new point whose x and y coordinates are obtained through modulo division of the coordinates of the given point by the divisor supplied by the caller.

multiply (multiplier) : {*nokia.maps.util.Point*}

This method creates a new point whose x and y coordinates are obtained by multiplying the coordinates of the given point by given multiplier.

round () : {*nokia.maps.util.Point*}

This method creates a new point whose x and y coordinates are obtained by rounding of the coordinates of the given point.

span (other) : {*nokia.maps.util.Point*}

here

Methods

This method creates a new point whose x and y coordinates represent the span (absolute delta) between the given point and the point supplied by the caller.

sub (other) : {*nokia.maps.util.Point*}

This method gets a new point by subtracting the point supplied by the caller from the given point.

toString () : {String}

This method creates a textual representation of the given point.

validate () : {Boolean}

This method checks if the x and y coordinates are valid numbers.

Class Description

This class represents a location within a two-dimensional space.

Constructor Details

nokia.maps.util.Point(x, y)

Parameters:

x:	{Number}
	The x coordinate of the location
y:	{Number}
	The y coordinate of the location

Property Details

static **isCoveredBy**: {String | Boolean}

This method checks if the x/y coordinate is covered by the given shape.

Method Details

add(other): {*nokia.maps.util.Point*}

This method obtains the sum of the given point and the point supplied by the caller.

Parameters:

other:	{ <i>nokia.maps.util.IPoint</i> }

here

A point which is to be added to the given point

Returns:

`{nokia.maps.util.Point}`

The new point representing a sum of the given point and that supplied by the caller

ceil(): `{nokia.maps.util.Point}`

This method creates a new point whose x and y coordinates are obtained by rounding up the coordinates of the given point.

Returns:

`{nokia.maps.util.Point}`

The rounded point

clone(): `{nokia.maps.util.Point}`

This method creates a exact clone of the given point.

Returns:

`{nokia.maps.util.Point}`

An object representing the cloned point

distance(other): `{Number}`

This method gets the distance to another point.

Parameters:

other: `{nokia.maps.util.IPoint}`

Another point

Returns:

`{Number}` The distance between the points

here

divide(divisor): {*nokia.maps.util.Point*}

This method creates a new point whose x and y coordinates are obtained by dividing the coordinates of the given points by given divisor.

Parameters:

divisor: {Number}
The divisor to use

Returns:

{*nokia.maps.util.Point*}
The point resulting from the division

equals(other): {Boolean}

This method checks if the given point is equal to the point supplied by the caller.

Parameters:

other: {*nokia.maps.util.IPoint*}
Another point to compare

Returns:

{Boolean} true if equal, otherwise false

floor(): {*nokia.maps.util.Point*}

This method creates a new point whose x and y coordinates are obtained by rounding down the coordinates of the given point.

Returns:

{*nokia.maps.util.Point*}
The new point

here

getNearest(first, second): {*nokia.maps.util.Point*}

This method calculates a point on the line defined by the two points provided by the caller such that calculated point lies nearest to the given point.

Parameters:

first: {*nokia.maps.util.IPoint*}

First point on the line

second: {*nokia.maps.util.IPoint*}

Second point on the line

Returns:

{*nokia.maps.util.Point*}

The point on the line that lies nearest to the given point

isCoveredBy(path, lineWidth, closed): {String | Boolean}

This method checks if the point is covered by the given polygon.

Parameters:

path: {Number[]}

The path of a polygon; the closing leg is not required

lineWidth: {Number}

The width of the line

closed: {Boolean} [optional, default: false]

A Boolean indicating if the path should be treated as closed (polygon)

Returns:

{String | Boolean}

false, if the point is not covered by the supplied polygon, otherwise a letter that indicates which part of the polygon covers the point:

- "V" a vertex
- "E" a edge

here

- "S" the surface

len(): {Number}

This method calculates the distance between the origin and the origin (x: 0, y: 0).

Returns:

{Number} The distance between the origin and the given point

max(other): {*nokia.maps.util.Point*}

This method creates a new point whose x and y coordinates are the largest values selected from the given point and the point supplied by the caller.

Parameters:

other: {*nokia.maps.util.IPoint*}
 Another point

Returns:

{*nokia.maps.util.Point*}
 The new point

min(other): {*nokia.maps.util.Point*}

This method creates a new point whose x and y coordinates are the smallest values selected from the given point and the point supplied by the caller.

Parameters:

other: {*nokia.maps.util.IPoint*}
 Another point

Returns:

{*nokia.maps.util.Point*}
 The new point

here

modulo(divisor): *{nokia.maps.util.Point}*

This method creates a new point whose x and y coordinates are obtained through modulo division of the coordinates of the given point by the divisor supplied by the caller.

Parameters:

divisor: {Number}
The divisor to use

Returns:

{nokia.maps.util.Point}
The new point

multiply(multiplier): *{nokia.maps.util.Point}*

This method creates a new point whose x and y coordinates are obtained by multiplying the coordinates of the given point by given multiplier.

Parameters:

multiplier: {Number}
The multiplier to use

Returns:

{nokia.maps.util.Point}
The point resulting from applying the multiplier to the given point

round(): *{nokia.maps.util.Point}*

This method creates a new point whose x and y coordinates are obtained by rounding of the coordinates of the given point.

Returns:

{nokia.maps.util.Point}

here

The new point

span(other): {*nokia.maps.util.Point*}

This method creates a new point whose x and y coordinates represent the span (absolute delta) between the given point and the point supplied by the caller.

Parameters:

other: {*nokia.maps.util.IPoint*}

Another point

Returns:

{*nokia.maps.util.Point*}

The new point

sub(other): {*nokia.maps.util.Point*}

This method gets a new point by subtracting the point supplied by the caller from the given point.

Parameters:

other: {*nokia.maps.util.IPoint*}

The point to subtract from the given point

Returns:

{*nokia.maps.util.Point*}

A point resulting from the subtraction

toString(): {String}

This method creates a textual representation of the given point.

Returns:

{String} The textual representation of the point

here

validate(): {Boolean}

This method checks if the x and y coordinates are valid numbers.

Returns:

{Boolean} true if the coordinates are valid, otherwise false.

Class: Rectangle

This class is a member of [nokia.maps.util](#).

Class Summary

This class represents a rectangular area in a two-dimensional coordinate space.

[For full details, see [nokia.maps.util.Rectangle](#)]

Table 280: Property Summary

Properties
bottomRight : { nokia.maps.util.Point } This property holds a Point object that contains the coordinates of the lower right corner of the given rectangle.
topLeft : { nokia.maps.util.Point } This property holds a Point object that contains the coordinates of the upper left corner of the given rectangle.

Table 281: Method Summary

Methods
addPoint (point) : { nokia.maps.util.Rectangle } This method computes a wider rectangle that covers the point specified by the caller.
addPoints (points) : { nokia.maps.util.Rectangle } This method widens the given rectangle so that it covers the points provided by the caller.
centerTo (point) : { nokia.maps.util.Rectangle } This method widens the rectangle so that the given point lies in its center.
clone () : { nokia.maps.util.Rectangle } This method obtains a clone of the given rectangle.

here

Class Description

This class represents a rectangular area in a two-dimensional coordinate space. The area is defined in terms of its top-left and bottom-right corners specified via instances of `nokia.maps.util.Point`.

Constructor Details

`nokia.maps.util.Rectangle`(points)

This method creates an instance of the class on the basis of the point objects provided by the caller.

Parameters:

points: {*nokia.maps.util.Point*[]}

A list containing at least one point; the the created rectangle contains all the received points

Throws:

{Error} Indicates that the points is empty.

Property Details

bottomRight: {*nokia.maps.util.Point*}

This property holds a `Point` object that contains the coordinates of the lower right corner of the given rectangle.

topLeft: {*nokia.maps.util.Point*}

This property holds a `Point` object that contains the coordinates of the upper left corner of the given rectangle.

Method Details

addPoint(point): {*nokia.maps.util.Rectangle*}

This method computes a wider rectangle that covers the point specified by the caller.

Parameters:

point: {*nokia.maps.util.Point*}

The point to cover

here

Returns:

{*nokia.maps.util.Rectangle*}

A reference to the newly computed rectangle

addPoints(points): {*nokia.maps.util.Rectangle*}

This method widens the given rectangle so that it covers the points provided by the caller.

Parameters:

points: {*nokia.maps.util.Point*[]}

An array of points to cover

Returns:

{*nokia.maps.util.Rectangle*}

A reference to the given rectangle

centerTo(point): {*nokia.maps.util.Rectangle*}

This method widens the rectangle so that the given point lies in its center.

Parameters:

point: {*nokia.maps.util.Point*}

The intended center of the rectangle

Returns:

{*nokia.maps.util.Rectangle*}

A reference to the given rectangle

clone(): {*nokia.maps.util.Rectangle*}

This method obtains a clone of the given rectangle.

Returns:

here

`{nokia.maps.util.Rectangle}`

A clone of the given rectangle

Class: Strip

This class is a member of [nokia.maps.util](#).

Class Summary

This class represents a special array that contains objects, all of the same type, and each with the same properties.

[For full details, see [nokia.maps.util.Strip](#)]

Table 282: Property Summary

Properties
data : {Object[]} This property holds the array values.
static constant LAT_LNG : { nokia.maps.util.Strip } This property is a template for ["latitude", "longitude"] strips.
names : {String[]} This property holds an array of the names of the properties defined on each of the objects in the strip.
offsets : {Object} This hashtable holds the relative offset of each object property held in the array (from 0 to n-1).
static constant X_Y : { nokia.maps.util.Strip } This property is a template for ["x", "y"] strips.

Table 283: Method Summary

Methods
static stencil (template, data) This method creates a new instance of Strip by reusing the meta data of an existing strip object and an optional data array.

Class Description

This class represents a special array that contains objects, all of the same type, and each with the same properties. The objects are not added to the array as objects, but instead the array contains just the values of the properties of the objects. Each object is allocated one slot in the array for every

here

property it contains. For example, if each object in the array has two properties called "x" and "y", then the length of the array is twice the number of objects added to it.

Constructor Details

nokia.maps.util.Strip(names, data)

This method creates a new instance of the class, using the arguments provided by the caller.

Parameters:

names: {String[]}

An array of strings holding the names of objects

data: {Object[]} [optional, default: []]

An array holding the property values for each of the objects identified in the to names argument

Property Details

data: {Object[]}

This property holds the array values. To get the number of objects contained in the array, the length of the data array must be divided by the length of the names property, so:

```
size = arr.data.length / arr.names.length;
```

static constant **LAT_LNG:** {*nokia.maps.util.Strip*}

This property is a template for ["latitude", "longitude"] strips.

names: {String[]}

This property holds an array of the names of the properties defined on each of the objects in the strip.

offsets: {Object}

This hashtable holds the relative offset of each object property held in the array (from 0 to n-1).

here

static constant **X_Y**: {[nokia.maps.util.Strip](#)}

This property is a template for ["x", "y"] strips.

Method Details

static **stencil**(template, data)

This method creates a new instance of `Strip` by reusing the meta data of an existing strip object and an optional data array.

Parameters:

template: {[nokia.maps.util.Strip](#)}

A `Strip` to be used as a template for creating a new instance of `Strip`

data: {Object[]} [optional, default: []]

An array of data with which to populate the new instance

Example:

```
strip1 = new Strip(["foo", "bar"], foobar1);
strip2 = Strip.stencil(strip1, foobar2);
```

Class: Vector3D

This class is a member of [nokia.maps.util](#).

Class Summary

This class represents a three dimensional vector.

[For full details, see [nokia.maps.util.Vector3D](#)]

Table 284: Property Summary

Properties	
readonly x : {Number}	This property holds the x component of the vector.
readonly y : {Number}	This property holds the y component of the vector
readonly z : {Number}	

here

Properties

This property holds the z component of the vector.

Table 285: Method Summary

Methods

add (other) : {*nokia.maps.util.Vector3D*}

This method creates the sum of the given vector and the vector provided by the caller.

angle (px, py, intertX, intertY) : {Number}

This method retrieves the angle of the given vector in decimal degrees to the given point within a coordinate system in which the highest (positive) values are in the top right.

divide (other) : {*nokia.maps.util.Vector3D*}

This method creates vector that represents the quotient obtained by dividing the given vector by the one supplied by the caller.

dot (other) : {Number}

This method creates the dot product of the given vector and the one provided by the caller.

magnitude () : {Number}

This method calculates the length of the given vector.

multiply (other) : {*nokia.maps.util.Vector3D*}

This method creates a vector that represents the product of the given vector and the one provided by the caller.

normal (other) : {*nokia.maps.util.Vector3D*}

This method creates the normal vector of the plane between the given vector and the one provided by the caller.

normalize () : {*nokia.maps.util.Vector3D*}

This method creates normalized representation of the given vector.

subtract (other) : {*nokia.maps.util.Vector3D*}

This method creates a vector that represents the difference between the given vector and the vector supplied by the caller.

Class Description

This class represents a three dimensional vector.

Property Details

readonly **x**: {Number}

This property holds the x component of the vector.

here

readonly **y**: {Number}

This property holds the y component of the vector

readonly **z**: {Number}

This property holds the z component of the vector.

Method Details

add(other): {*nokia.maps.util.Vector3D*}

This method creates the sum of the given vector and the vector provided by the caller.

Parameters:

other: {*nokia.maps.util.Vector3D*}

The vector to be added

Returns:

{*nokia.maps.util.Vector3D*}

A vector representing the sum of the given vector and the vector supplied by the caller

angle(px, py, intertX, intertY): {Number}

This method retrieves the angle of the given vector in decimal degrees to the given point within a coordinate system in which the highest (positive) values are in the top right.

For screen coordinates, the y-axis must be inverted. For a normalized vector, the method can be called without parameters to return its normalized angle.

Parameters:

px: {Number} [optional, default: 0]

The x-coordinate of the point to which the angle is to be calculated; if not provided zero is used

py: {Number} [optional, default: 0]

here

The y-coordinate of the point to which the angle is to be calculated; if not provided zero is used

intertX: {Boolean} [optional, default: false]

If provided and `true`, the x-axis is inverted, therefore the coordinates are increasing to the left and decreasing to the right

intertY: {Boolean} [optional, default: false]

If provided and `true`, the y-axis is inverted, therefore the coordinates are increasing to the bottom and decreasing to the top

Returns:

{Number} The angle of this vector in degrees, relative to the given coordinate

divide(other): {*nokia.maps.util.Vector3D*}

This method creates vector that represents the quotient obtained by dividing the given vector by the one supplied by the caller.

Parameters:

other: {*nokia.maps.util.Vector3D*}

The vector by which divide the given vector

Returns:

{*nokia.maps.util.Vector3D*}

A vector representing a quotient of the division of the given vector by the one supplied by the caller

dot(other): {Number}

This method creates the dot product of the given vector and the one provided by the caller.

Parameters:

other: {*nokia.maps.util.Vector3D*}

here

The vector with which the given vector is to be dot-multiplied

Returns:

{Number} The dot product of the two vectors

magnitude(): {Number}

This method calculates the length of the given vector.

Returns:

{Number} The length of this vector

multiply(other): {*nokia.maps.util.Vector3D*}

This method creates a vector that represents the product of the given vector and the one provided by the caller.

Parameters:

other: {*nokia.maps.util.Vector3D*}

The vector to be multiplied

Returns:

{*nokia.maps.util.Vector3D*}

A vector resulting from the multiplication of the given vector by the one supplied by the caller

normal(other): {*nokia.maps.util.Vector3D*}

This method creates the normal vector of the plane between the given vector and the one provided by the caller.

Parameters:

other: {*nokia.maps.util.Vector3D*}

The vector provided by the caller

here

Returns:

`{nokia.maps.util.Vector3D}`

The normal vector of the plane between the given vector and that provided by the caller

normalize(): `{nokia.maps.util.Vector3D}`

This method creates normalized representation of the given vector.

Returns:

`{nokia.maps.util.Vector3D}`

The normalized representation (length = 1) of the vector

subtract(other): `{nokia.maps.util.Vector3D}`

This method creates a vector that represents the difference between the given vector and the vector supplied by the caller.

Parameters:

other: `{nokia.maps.util.Vector3D}`

The vector to be subtracted from the given vector

Returns:

`{nokia.maps.util.Vector3D}`

A vector resulting from the subtraction

here