HERE Maps API - How to interact with Markers

This article explains how to set up a map event `listener` so that designated markers on the map are **interactive**. If a marker is given an `$href` attribute, clicking on the marker will forward the user to the given URL. If a marker is given a `$click` attribute, the browser will run the text of `$click` attribute as JavaScript when the marker is clicked. Through a combination of these functions most map interactions should be satisfied. This is an example of *event delegation*, since the `click` events of many map objects are serviced but only one `click` event listener is created.

Introduction

A frequent use case of the Nokia Map API, would be where data is summarised by displaying it on a map, and **clicking** on the map markers should result in further information being displayed about that particular marker. This **interaction** can be achieved through associating some extra data with each marker and adding an event listener to read and process that data. The simple JavaScript functions described below can be re-used on an web page to process standard `onclick` and forward events

Code Commentary

Adding a "click" Event Listener to the Map

Tip: If you have lots of groups of map objects and want each subset to do different things, you could place each group into its own map `container` and wire up the event listener on each individual map `container` instead.

After having created a `map` in the usual manner, a `click` event listener needs to be created. This will fire whenever the map itself **or any of its children** is clicked. To reduce the interaction down to only to certain objects, we can check the properties of the `evt.target` (i.e. the actual object that was clicked) and see if it satisfies our criteria. In this case we want to service two cases :

- If a marker has an `$href` attribute, clicking on the marker will forward the user to the given URL.
- If a marker has a `$click` attribute, the browser will run the text of `$click` attribute as JavaScript when the marker is clicked.

Hence there is a check to see if the given attribute has been defined, followed by the necessary code to service the request. To forward a browser to another web page, we simply need to set the `window.location`; to run some arbitrary JavaScript, we could use the `eval()` method, but it is better practice to create our own function as shown and to run it directly.

```
map.addListener("click", function(evt) {
 if (( evt.target.$href === undefined) == false){
    window.location = evt.target.$href;
 }  else if (( evt.target.$click === undefined) == false){
    var onClickDo = new Function(evt.target.$click);
  onClickDo();
  }
});
```

Adding Visual Feedback

Markers give points of focus to a map, but there is nothing intrinsic about them which means that a user will automatically click on them. You need to give the user some sort of visual indication that the markers are to be clicked. A lack of visual feedback (especially when the interaction results in updating the page) is known as "Mystery Meat Navigation" and results in a poor user experience. It is much better Human Computer Interaction to **warn** the user prior to his decision that something will happen if they click on the screen at this point. We need to change the shape of the mouse pointer when the user hovers over an interactive marker. This can be done by dynamically altering the `document.body.style.cursor` style.

```
function changeCursor(target, cursor){
 if ((( target.$href === undefined) == false) ||
    (( target.$click === undefined) == false)){
  document.body.style.cursor = cursor;
  }
```


(C) Copyright Nokia 2014. All rights reserved.

```
}
```

Again we can wire up the `click`, `mouseover` and `mouseout` events to ensure that the mouse pointer is altered when a relevant marker has focus, and changed back when that focus is lost (or the marker has been clicked.)

```
map.addListener("click", function(evt) {
 changeCursor(evt.target, 'default');
.... etc ...
});

map.addListener("mouseover", function(evt) {
   changeCursor(evt.target, 'pointer');
});

map.addListener("mouseout", function(evt) {
 changeCursor(evt.target, 'default');
});
```

Creating the markers

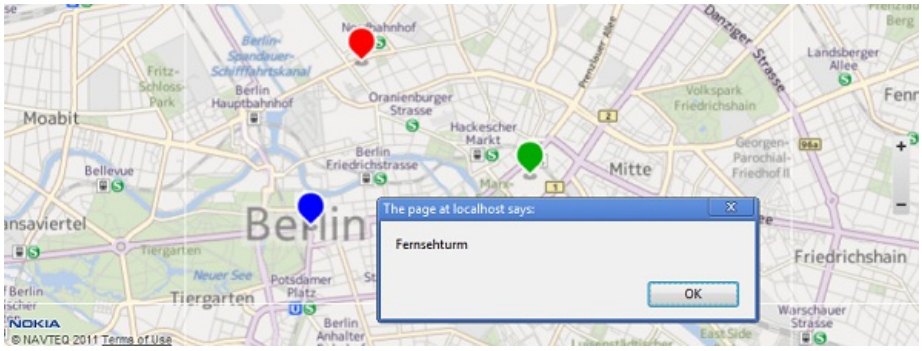
The markers are created in the usual manner, but each **interactive** marker has an additional attribute. A marker with an `$href` attribute works like a link.

```
var markerWithLink= new nokia.maps.map.StandardMarker(
 new nokia.maps.geo.Coordinate(52.516237, 13.377686),
 {$href  : 'http://example.com/'}
);
```

A marker with a `$click` attribute and works like a firing an `onclick` event.

```
var markerWithClickEvent= new nokia.maps.map.StandardMarker(
 new nokia.maps.geo.Coordinate(52.520816, 13.409417),
 {$click  : 'alert("Hello World");'}
);
```

Summary



A working example which bundles the interactivity into a map component and shows **interactive** markers in action can be found at:

http://heremaps.github.io/examples/examples.html#marker-interactivity-component__index

- The Blue marker works like a link, clicking on it forwards you to a new page
- The Green marker executes a line of JavaScript (as shown) when it is clicked.
- The Red marker is just a marker, it does nothing special, note that cursor does not change as you hover over it.

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.