HERE Maps API - How to create a Tooltip

This article explains how to use CSS styling to create an **tooltip**. It is an example of the use of the `mouseover` and `mouseout` events.

Introduction

A **tooltip** is a GUI element that appears when the cursor hovers over an associated point of interest. It usually contains some text with more information about the item concerned, typically a short descriptive label. This leaves the screen uncluttered by hiding these summaries unless the point of interest has focus. It is impossible to click on a tooltip, since the tooltip will disappear as soon as its associated point of interest loses focus. A more common clickable alternative to the tooltip is the `InfoBubble`, which is part of the standard API. Since tooltips are only used on a subset of mapping applications, they do not come as standard as part of the HERE Maps API, if tooltip functionality is required in an application, it is a simple matter to add tooltips to map markers with a few lines of code.

Coded Example

In HTML, a **tooltip** is simply an extra `<div>` element, which is dynamically styled to appear or disappear as associated markers gain or lose focus.

1) Inject an extra DIV into the DOM

Assume the web page holds a map in the following `<div>` element

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

After setting up the map in the usual manner, we need to inject an additional `<div>` into the DOM - this will eventually hold the text of the **tooltip**.

```
var node = document.createElement("div");
node.className = 'nm_bubble_content';
node.id = 'nm_tooltip';
document.getElementById("mapContainer").appendChild(node);
```

The `<div>` element above has been given a unique `id`, and therefore can be styled directly using CSS:

```
#nm_tooltip{
      position: absolute;
      color:blue;
      background:white;
      border: 1px solid black;
      padding-left: 1em;
      padding-right: 1em;
      display: none;
      min-width: 120px;
}
```

The three important CSS attributes here are `background`, `color` and `display`. A fixed `background` colour needs to be set so that the tooltip itself is displayed over the map, `color` will need to contrast with the `background` of course. The `display` **must** initially be set to `none` so that the tooltip `<div>` element is initially invisible. The other CSS style attributes such as `padding` and `border` can be altered as you see fit. It is also possible to style the `<div>` using its style class.

2) Add an "onHover" effect by handling mouseover and mouseout events.

We want to display the tooltip `<div>` whenever a map object which has an associated `$tooltip` attribute gains focus. In order to avoid writing multiple event listeners, we will add a listener to the map itself, and check which object has triggered the event within the listener function. Hiding and displaying the tooltip is only the relevant for objects with a `$tooltip` attribute, and the presence of this can be determined through using a `undefined` check The tooltip is displayed by setting its `style` to `display:block`.

```
map.addListener("mouseover", function(evt) {
 if (( evt.target.$tooltip === undefined) == false){
  document.getElementById("nm_tooltip").style.display ='block';
  ..etc...
 }
});
```

Similarly the tooltip `<div>` can be hidden whenever an object with a `$tooltip` attribute loses focus. Optionally a third listener for `click` events could be added here to hide the tooltip if an `InfoBubble` is displayed for example.

```
map.addListener("mouseout", function(evt) {
 if (( evt.target.$tooltip === undefined) == false){
  document.getElementById("nm_tooltip").style.display ='none';
 }
});
```

If the map is *draggable*, and the tooltip is displayed whilst doing this, the **tooltip** will also need to move as the map is moved. This can be achieved by re-firing the marker's `mouseover` event

```
map.addListener("drag", function(evt) {
  if (document.getElementById("nm_tooltip").style.display == 'block'){
       map.dispatch( new nokia.maps.dom.Event({type: "mouseover", target: map.getObjectAt(evt.displayX, evt.displayY)}));
  }
});
```

3) Add a function to move the tooltip to the correct location and add the relevant text.

Further styling is required on the tooltip `<div>` to move it around the map. The bounding box of a marker can be obtained using the `target.getDisplayBoundingBoxl()` method and extracting the x and y coordinates. The x co-ordinate is further offset by half of the width of the `<div>` so that the centre of the tooltip is aligned with the marker. The text is added by setting the `innerHTML` attribute. Note that it is inefficient to keep traversing the DOM, so a temporary variable is assigned to the DOM node holding the tooltip `<div>`, to avoid searching for it more than once.

```
map.addListener("mouseover", function(evt) {
 if (( evt.target.$tooltip === undefined) == false){
  var tooltip = document.getElementById("nm_tooltip");
  var target = evt.target;
  tooltip.innerHTML =  target.$tooltip;
  tooltip.style.display ='block';
  tooltip.style.left = target.getDisplayBoundingBox(map).getCenter().x - (tooltip.offsetWidth/2);
  tooltip.style.top =  target.getDisplayBoundingBox(map).bottomRight.y + 1;
 }
);
```

4) Define a Marker with a tooltip and add it to the map

Having made all the necessary preparations, a `StandardMarker` can be added in the usual way. The additional `$tooltip` attribute will contain the associated tooltip text.

```
var brandenburgerTorMarker = new nokia.maps.map.StandardMarker(
 new nokia.maps.geo.Coordinate(52.516237, 13.377686),
 {$tooltip : "Brandenburger Tor"}
);
```

Because the tooltip text is displayed within an HTML `<div>` element, it is also able to accept formatted HTML as shown:

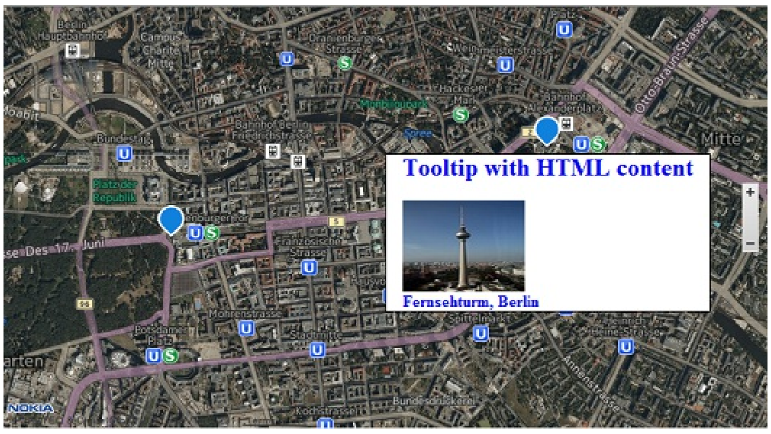
```
var fernsehturmTooltip = '<div>' +
  '<h2>Tooltip with HTML content</h2>' +
  '<img width=120 height=90 src=' +
  '"http://upload.wikimedia.org/wikipedia/commons/' +
  '8/84/Berlin-fernsehturm.JPG" ' +
  'alt=""/><br/><b>Fernsehturm, Berlin</b>' +
    '</div>';
var fernsehturmMarker = new nokia.maps.map.StandardMarker(
 new nokia.maps.geo.Coordinate(52.520816, 13.409417),
 {$tooltip : fernsehturmTooltip}
);
```

Obviously the injected HTML can then be further styled as required.

Summary

A working example which bundles the tooltips into a map component can be seen below, the cursor is hovering over the marker on the right.

http://rawgithub.com/heremaps/examples/master/maps_api_for_javascript/demos/tooltip-component/index.html


Tooltip with HTML content
Fernsehturm, Berlin
Mitte
NOKIA