Google Mapls API - Getting Started

These notes are based (with a few changes) on the Google Maps JavaScript tutorial which can be found at:

https://developers.google.com/maps/documentation/javascript/tutorial

This documentation is designed to let you quickly start exploring and developing applications with the Google Maps API. It contains slightly more detail than the original Google tutorial.

Hello, World

The easiest way to start learning about the Google Maps API is to see a simple example. The following web page displays a map centered on Sainsbury's, Thornhill, Cardiff:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">

    <!--
    Always force latest IE rendering engine (even in intranet) & Chrome
    Frame. Remove this if you use the .htaccess
    -->
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

    <title>Google Maps</title>
    <meta name="description" content="google maps api test">
    <meta name="author" content="Peter">

    <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />

    <!--
    Explicitly set the size of everything, in case the browser is in quirks
    mode. In quirks mode, the map size would be set to 0,0
    -->
    <style type="text/css">
      html {
        height: 100%
      }
      body {
        height: 100%;
        margin: 0;
        padding: 0
      }
      #map_canvas {
        height: 100%
      }
    </style>

    <!-- Fetch the maps API and set sensor to FALSE (not using GPS etc) -->
    <script type="text/javascript"
    src="http://maps.googleapis.com/maps/api/js?sensor=FALSE"></script>

    <!--
    Declare and define a function called 'initialize'. The body of this
    function defines the map options in a variable called 'mapOptions', and
    then creates a new Map object in the variable 'map', giving it the
```

```
    'map_canvas' element along with the map options in 'mapOptions'.
  -->
  <script type="text/javascript">
    function initialize() {
      // latitude and longitude can be obtained from
      // http://universimmedia.pagesperso-orange.fr/geo/loc.htm

      var theCenter = new google.maps.LatLng(51.53884, -3.193194);
      var mapOptions = {
        center : theCenter,
        zoom : 16,
        mapTypeId : google.maps.MapTypeId.ROADMAP,
        streetViewControl : true
       };

       // Fill the 'map_canvas' div
       // NOTE: this will only work if you have already included the maps API
       var map = new google.maps.Map(
                document.getElementById("map_canvas"), mapOptions);

      // Add a marker
       var marker = new google.maps.Marker({
        position : theCenter,
        map : map,
        title : "Sainsbury's Thornhill"
       });

      // Add an InfoWindow...
      // The text to go inside the window
      var contentString = '<div id="content">' +
              '<div id="infotitle"></div>' +
              '<h1>Sainsburys Superstore</h1>' +
              '<div id="infoContent">' +
                '<p><em>Sainsburys Superstore</em> is located near ' +
                'N. Cardiff Medical Centre ' +
                'on Excalibur Drive, Thornhill.</p>' +
              '</div>' +
            '</div>';

      // Create the InfoWindow object (note: this doesn't display it)
      var infowindow = new google.maps.InfoWindow({content: contentString});

      // Add an event listener so that the info window is
      // shown when the marker is clicked
      google.maps.event.addListener(marker,
                                    'click',
                                    function() {
                                       infowindow.open(map, marker);
                                    });

    }
  </script>

</head>

<!-- Call the initialize function defined above when the page body loads -->
<body onload="initialize()">

  <!-- define a div with id 'map_canvas' to  hold the map -->
  <div id="map_canvas" style="width:100%; height:100%"></div>

</body>
```

```
</html>
```

Even in this simple example, there are a few things to note:

1. We declare the application as HTML5 using the `<!DOCTYPE html>` declaration.
2. We include the Maps API JavaScript using a `script` tag.
3. We create a `div` element named "map-canvas" to hold the Map.
4. We create a JavaScript object literal to hold a number of map properties.
5. We create a JavaScript "map" object, passing it the `div` element along with the map properties.
6. We use an event listener to load the map after the page has loaded.

These steps are explained below.

Declaring Your Application as HTML5

We recommend that you declare a true `DOCTYPE` within your web application. Within the examples here, we've declared our applications as HTML5 using the simple HTML5 `DOCTYPE` as shown below:

```
<!DOCTYPE html>
```

Most current browsers will render content that is declared with this `DOCTYPE` in "standards mode" which means that your application should be more cross-browser compliant. The `DOCTYPE` is also designed to degrade gracefully; browsers that don't understand it will ignore it, and use "quirks mode" to display their content.

Note that some CSS that works within quirks mode is not valid in standards mode. In specific, all percentage-based sizes must inherit from parent block elements, and if any of those ancestors fail to specify a size, they are assumed to be sized at 0 x 0 pixels. For that reason, we include the following `<style>` declaration:

```
<style type="text/css">
  html { height: 100% }
  body { height: 100%; margin: 0; padding: 0 }
  #map-canvas { height: 100% }
</style>
```

This CSS declaration indicates that the map container `<div>` (named `map-canvas`) should take up 100% of the height of the HTML body. Note that we must specifically declare those percentages for `<body>` and `<html>` as well.

Loading the Google Maps API

```
<html>
  <head>
    <script type="text/javascript"
       src="http://maps.googleapis.com/maps/api/js?sensor=FALSE"></script>
```

The URL contained in the `script` tag is the location of a JavaScript file that loads all of the symbols and definitions you need for using the Google Maps API. This `script` tag is required.

Asynchronously Loading the API

You may wish to load the Maps API JavaScript code after your page has finished loading, or on demand. To do so, you can inject your own `<script>` tag in response to a `window.onload` event or a function call, but you need to additionally instruct the Maps JavaScript API bootstrap to delay execution of your application code until the Maps JavaScript API code is fully loaded. You may do so using the `callback` parameter, which takes as an argument the function to execute upon completing loading the API.

The following code instructs the application to load the Maps API after the page has fully loaded (using `window.onload`) and write the Maps JavaScript API into a `<script>` tag within the page. Additionally, we instruct the API to only execute the `initialize()` function after the API has fully loaded by passing `callback=initialize` to the Maps API bootstrap:

```
function initialize() {
  var mapOptions = {
    zoom: 8,
    center: new google.maps.LatLng(-34.397, 150.644)
  };
  var map = new google.maps.Map(document.getElementById('map-canvas'),
      mapOptions);
}
function loadScript() {
  var script = document.createElement('script');
  script.type = 'text/javascript';
  script.src = 'https://maps.googleapis.com/maps/api/js?v=3.exp&' +
      'callback=initialize';
  document.body.appendChild(script);
}
window.onload = loadScript;
```

Map DOM Elements

```
<div id="map-canvas" style="width: 100%; height: 100%"></div>
```

For the map to display on a web page, we must reserve a spot for it. Commonly, we do this by creating a named `div` element and obtaining a reference to this element in the browser's document object model (DOM).

In the example above, we define a `<div>` named "map-canvas" and set its size using style attributes. Note that this size is set to "100%" which will expand to fit the size on mobile devices. You may need to adjust these values based on the browser's screensize and padding. Note that the map will always take its size from that of its containing element, so you must always set a size on that `<div>` explicitly.

Map Options

```
var mapOptions = {
  center: new google.maps.LatLng(-34.397, 150.644),
  zoom: 8
};
```

To initialize a Map, we first create a *`Map options`* object to contain map initialization variables. This object is not constructed; instead it is created as an object literal. There are two required

options for every map: `center` and `zoom`.

Latitudes and Longitudes

Because we want to `center` the map on a specific point, we create a `LatLng` object to hold this location by passing the location's coordinates in the order: latitude, longitude:

```
var theCenter = new google.maps.LatLng(51.53884, -3.193194);
var mapOptions = {
   center : theCenter,
```

To find the latitude and longitude of a location, you can use this site:

http://universimmedia.pagesperso-orange.fr/geo/loc.htm

Zoom Levels

The initial resolution at which to display the map is set by the `zoom` property, where zoom `0` corresponds to a map of the Earth fully zoomed out, and higher zoom levels zoom in at a higher resolution.

```
zoom: 8
```

Offering a map of the entire Earth as a single image would either require an immense map, or a small map with very low resolution. As a result, map images within Google Maps and the Maps API are broken up into map "tiles" and "zoom levels." At low zoom levels, a small set of map tiles covers a wide area; at higher zoom levels, the tiles are of higher resolution and cover a smaller area.

The Map Object

```
var map = new google.maps.Map(document.getElementById("map-canvas"),
    mapOptions);
```

The JavaScript class that represents a map is the `Map` class. Objects of this class define a single map on a page. (You may create more than one instance of this class - each object will define a separate map on the page.) We create a new instance of this class using the JavaScript `new` operator.

When you create a new map instance, you specify a `<div>` HTML element in the page as a container for the map. HTML nodes are children of the JavaScript `document` object, and we obtain a reference to this element via the `document.getElementById()` method.

This code defines a variable (named `map`) and assigns that variable to a new `Map` object, also passing in options defined within the `mapOptions` object literal. These options will be used to initialize the map's properties. The function `Map()` is known as a *constructor* and its definition is shown below:

Constructor	Description
`Map(mapDiv:Node, opts?:`*`MapOptions`*`)`	Create a new map in the specified HTML container, usually a DIV, according to the specified options, if any are given.

Loading the Map

While an HTML page renders, the document object model (DOM) is built out, and any external images and scripts are received and incorporated into the `document` object. To ensure that our map is placed on the page after the page has fully loaded, we only execute the function which constructs the `Map` object once the `<body>` element of the HTML page receives an `onload` event. Doing so avoids unpredictable behaviour and gives us more control on how and when the map is drawn.

The Google Maps JavaScript API provides a set of events that you can handle to determine state changes. For more information, see Map Events.

```
google.maps.event.addDomListener(window, 'load', initialize);
```

Alternatively, you can use the `body` tag's `onload` attribute.

```
<body onload="initialize()">
```