

Week 8 – Google Maps

Introduction

Hopefully by now you'll have seen the possibilities that jQuery provides for rich content on web sites in the form of interaction and media playback. This week we'll be extending this into the realm of Google Maps.

Google makes it easy to embed a map into a web page, but Google also provides an API (Application Programming Interface) for web designers wanting to do more advanced work with maps.

This week you'll learn how to embed a Google Map into a web page and add custom markers with custom labels.

Setting Up

The Google Map API provides access via a number of programming languages. Its JavaScript access is highly comprehensive but requires knowledge of 'vanilla' JavaScript. Luckily, there is a useful library that allows us to interact with the API using jQuery. In order to make use of this we have to do a bit of setting up and will need to include the following into our HTML document:

- jQuery (you should already have this in your folder)
- Google Maps JavaScript API (we'll pull this off Google's servers)
- jQuery Map library (available for download via the MMCC2041 page, place in the same folder as your HTML files)

The <head> section of the default HTML document (available from the MMCC2041 page) looks like this:

```
<head>
<meta charset="utf-8">
<title>MMCC2041</title>
<link rel="stylesheet" href="style.css">
<script src="https://maps.google.com/maps/api/js?key=
AIzaSyClkwdCPHCWJj-OUiKT1bwazL37BXS6sqg"></script>
<script src="jquery-3.6.0.min.js"></script>
<script src="jquery.ui.map.full.min.js"></script>
<script>
$(document).ready(function() { // do not delete this line

}); // do not delete this line
</script>
</head>
```

You can see (in bold) that we've added a couple of extra lines to include a) the Google Maps API and b) the jQuery Maps library. *Make sure that these <script> tags are in the same order as above or things won't work properly.*

Adding a Map

There are three steps to adding a map to your web page. First, you must code some HTML. Second, you can add some CSS to style the map (add a border, position on the page). Third, there's some hefty jQuery required to add markers and labels. We'll approach each in turn.

Map HTML

Adding the HTML is the easiest part. All the HTML does is create a space on the page in which the map will load. We can use an empty `<div>` for this:

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

Don't forget to give the `<div>` an ID. We'll use the ID as a hook for the jQuery.

Ok, that's the HTML dealt with – don't get complacent, it gets more difficult later!

Map CSS

We'll create a CSS rule to set the height and width of the map. We'll also add a couple of extra bits to give it a border and centre it on the page.

```
#myMap {  
    width: 800px;  
    height: 450px;  
    border: 3px solid #000000;  
    margin: auto;  
}
```

There's nothing in there that should be new to you, but if you have any questions just ask. You can set the width and height to whatever you want, I've just chosen these values arbitrarily.

If you check your HTML page in a web browser you should currently just see a rectangle with black border that's centred on the page. Next we'll get a map to load in that space.

Map jQuery

Ok, this first bit will look easy, but trust me, it'll get more complex.

If you downloaded the default HTML document from the MMCC2041 page then you'll already have the `$(document).ready(function() { });` coded in place. Insert the following code in the place where we usually put our jQuery:

```
$( '#myMap' ).gmap( );
```

In brief, this line targets our `<div>` with the ID `myMap` and tells the maps library that this is where the map will be loaded. Save your HTML file and test it in the browser. You should see a map loaded into the `<div>`.

If your map isn't loading then check that you have all the `<script>` tags in the right place and that you have downloaded the jQuery and jQuery maps libraries to the same folder as your HTML document.

Now that we've got a map on the page we can now start to add to the code to achieve the following using *options*:

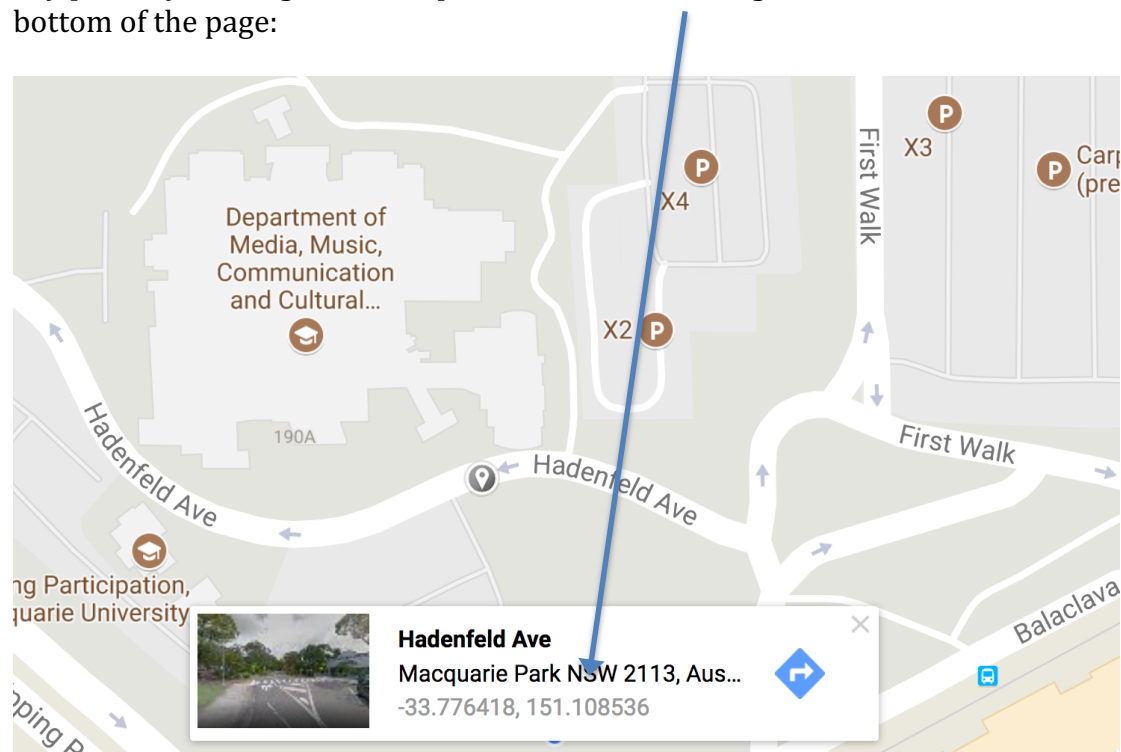
- Centre the map on a specified location
- Zoom in to a specified level
- Set the type of map (roadmap, satellite, hybrid or terrain)

The following options go inside the parenthesis of `.gmap()` and because we're using multiple options we need to place them inside of `{ }` just as we do with `.css()` and `.animate()`. Let's add them one at a time so you can see the changes in the HTML document. I'll space them out over individual lines so it's easier to read and follow.

```
$( '#myMap' ).gmap( {  
    'center': '-33.773422,151.11267'  
});
```

Option: 'Center'

The 'center' option (note US English spelling) uses the latitude and longitude coordinates to set the centre point for the map. In the above example, these are the coordinates for Macquarie University. You can use find the coordinates of any place by clicking on the map – the latitude and longitude will be visible at the bottom of the page:



You can try putting in your own address and replacing the coordinates in my example with your own.

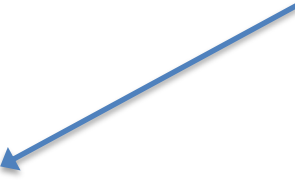
Test your HTML document in the browser.

Option: 'Zoom'

At the moment we're using the default zoom setting, which I think it about 5. Zoom values range from 0 – 25 (depending on the location you're viewing, some places only go up to 19).

Add this code on the line after the 'centre' option. Remember that now as we'll have more than one option, we need to separate them with a comma, but the last entry doesn't have a trailing comma:

```
$( '#myMap' ).gmap( {  
    'center': '-33.773422,151.11267',  
    'zoom': 17  
});
```



Test your HTML document in the browser.

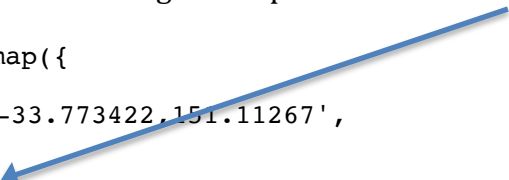
Option: 'MapTypeId'

As mentioned above, there are four maps types available:

- ROADMAP
- SATELLITE
- HYBRID
- TERRAIN

We can set the type of map using the 'MapTypeId' option. Add this after the 'zoom' option, and don't forget to separate with a comma:

```
$( '#myMap' ).gmap( {  
    'center': '-33.773422,151.11267',  
    'zoom': 17,  
    'MapTypeId': google.maps.MapTypeId.SATELLITE  
});
```



All map type values are set using `google.maps.MapTypeId.TYPE-OF-MAP`. The type must be in uppercase (see the four types above).

Test your HTML document in the browser and try changing to different map types.

Summary so Far

Ok, so far so good. You've learned how to insert a Google map onto a page and set the centre point, zoom level and map type.

Next, we'll look at how to add custom markers and labels.

Adding Custom Markers

Adding custom markers and labels is quite easy but the code is quite verbose so pay attention to detail.

To add a marker we have to tell the browser where to place it so we'll need latitude and longitude coordinates again. You should use the same ones from when we set the centre point above.

Note, that what we're about to code goes outside of the code to insert the map on the page, i.e. it goes after the final `})` but before the `})` that closes the `document.ready()` function.

Add the following code:

```
$('#myMap').gmap('addMarker',{ 'position': '-33.773422,151.11267' });
```

So again, we're targeting the <div> holding the map using a jQuery selector and we're telling to add a marker at specific coordinates.

Test your HTML document in the browser and you should see a marker on the map.

Try clicking the marker, it doesn't do anything, so next we'll add a `click()` action to the marker and get it to display a label.

Adding Custom Labels

You may recall from earlier jQuery classes that you can "chain" jQuery commands together, for example:

```
$('#myDiv').fadeOut().fadeIn().fadeOut().fadeIn();
```

We're going to "chain" the `.click()` action onto the end of what we coded in the previous step.

```
$('#myMap').gmap('addMarker',{ 'position': '-33.773422,151.11267' })  
.click(function() { });
```



More code to come in this space!

Note, I've placed the `.click()` action on the following line. This is fine and will still work. It just makes it a bit easier to follow.

It's worth pointing out at this stage that just like with any click event using jQuery we can make anything happen. For example, you could start playback of a video or audio media, go to another web page, or make an image or other block of text slide in, fade in, animate or appear on the page. Today, however, we're going to use the Google Maps API to make a label appear above the marker.

In the space indicated above we'll add the following code. This is the action performed by the function called when the marker is clicked:

```
$('#myMap').gmap('openInfoWindow', { 'content': '<p>Text Goes  
Here</p>' }, this);
```

The 'content' option accepts HTML so you can encode a hyperlink or an image in there if appropriate. The complete code for this section should look like this:

```
$('#myMap').gmap('addMarker',{ 'position': '-33.773422,151.11267' })  
.click(function() {  
    $('#myMap').gmap('openInfoWindow', { 'content': '<p>Text Goes  
Here</p>' }, this);  
});
```

By replicating the above code with different latitude and longitude coordinates, you can of course have multiple markers on a map.

Summary

This is quite complex compared to what we've previously covered but it does provide possibilities for a sophisticated level of engagement.