SEM2220: Mobile Web Assignment

Samuel Jackson
University Of Aberystwyth
slj11@aber.ac.uk

February 24, 2016

1 Introduction

This report provides the documentation for the mobile web assignment for SEM2220. The aim of this assignment is to add some additional functionality to an existing code base comprising of a responsive website wrapped as a PhoneGap application for deployment as a native mobile application. Section 2 discusses the implementation of a dynamic sessions lists as specified in the assignment brief. This section will discuss both the final solution and any problems encountered. Section 3 discusses an additional feature implemented for "flair" marks. Again this will focus on the solution produced and outline any difficulties encountered along the way.

Before beginning to implement either of these two parts I first took the existing code base and converted it into a PhoneGap application, following the steps outlined during the practical workshops. I also wrote a small script that builds locally and deploys the project to an Android device connected via USB. This enabled me to rapidly deploy the application to a real device for testing in both of the following sections. The full PhoneGap project, including the deployment script and the source for this document are supplied in the zip folder submission.

2 Implementing the Dynamic Sessions List

Implementing the dynamic sessions list involved interpreting the existing code base and then implementing several steps to produce a working sessions list. The existing code base already implemented the basic functionality of the website, including navigation between pages and setting up a local database with test data.

There were two main areas that needed to be extended in order to get dynamic sessions loading working proper which were marked with inline comments. The first of these was in the *Controller.js* file in a function called *renderSessionsList.* This function acts as an event handler which responds to sessions being loaded from the local database. It's purpose is to render the data received from the database query as HTML code. This function should be concerned with how sessions are presented to the user.

The second place that required changes in order for the code to work was in the *DataContext.js* file. This file is responsible for controlling operations on the local database. All that was required in this file was to implement the function *querySessions* to retrieve session data from the database.

The code in the *DataContext.js* file is possibly the simplest code modification. It executes an SQL select statement to pull the session data from the table in the local database for the first day (as indicated by the comments). The response is then passed to a handler function called *handleSessionsLoaded.* This performs two duties. Firstly it converts the loaded *SQLResultSet* to a vanilla Javascript array. I chose to do this so that the presentation code in *Controller.js* could use generic array manipulation functions (e.g. *map, forEach, reduce*) which are not present on the result object. This also hides the format of the underlying data storage system resulting in better decoupling. Its second duty is to call the *processorFunc* with the query result (now as an array). This *processorFunc* is set to the event handler *renderSessionsList*, yielding control back to the *Controller.js.*

The *processorFunc* in the *Controller.js* file converts the returned array of session data into HTML elements which match the format of the original static structure of the sessions list provided in the module tutorials. The resulting HTML elements are then inserted into the DOM using the jQuery library. Once the new HTML content has been inserted into the DOM jQuery Mobile (JQM) event handlers must be bound to the new content. According to the JQM documentation [1] newly created components need to be initialised by calling *.trigger('create')* on the parent HTML element of the newly inserted HTML.

Finally, a new filterable widget is added to the content of the page. This is created after the listview widget has been initialised so that the events and styling of the filter widget are setup correctly. One of the issues I encountered when implementing this part of the project was that widgets which were not created or setup correctly caused either the functionality to break or weird CSS styling (specifically extra padding above the filter to appear).

3 Implementing Interactive Venue Maps

For the additional "flair" marks for this assignment I chose to implement interactive google maps in the project. This feature loads the venues for the current day from the local database and shows them as markers on the map. Clicking/tapping on the marker will cause an info window to appear with the name of the location shown.

There are several parts to this feature. Firstly, instead of generating a static map from the Google maps API I imported the full JavaScript library inside the *index.html* file. This provides a much richer API for creating maps which are both more pleasant for the user and interactive, resulting in a much richer UI experience.

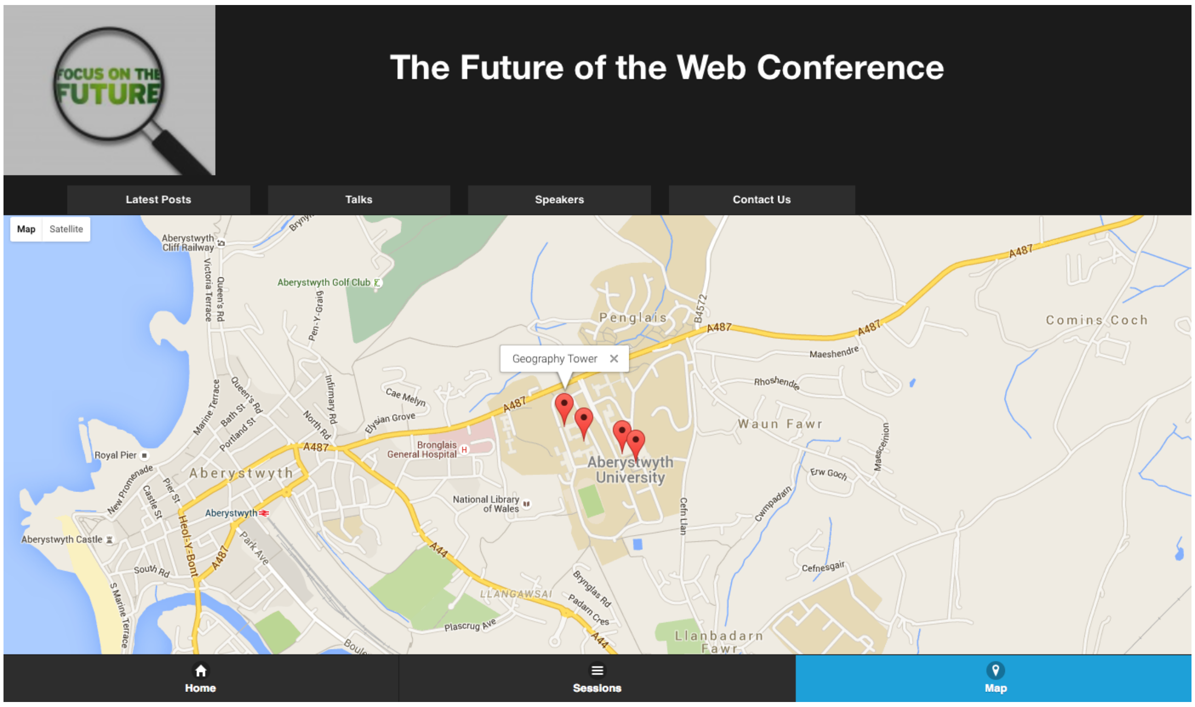


Figure 3.1: *Screen capture showing the maps page with markers for the venues and an info window showing the location name*

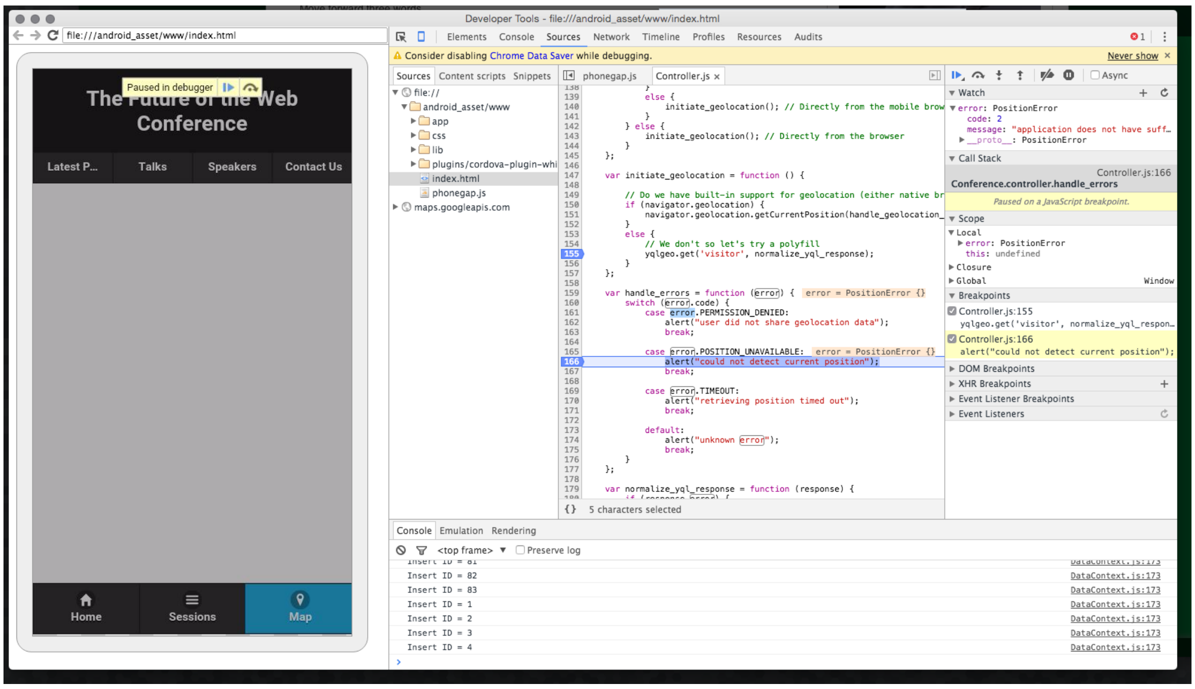
The next part of this feature required me to load the venues from the local database when the user clicks on the map page so that they can be displayed as markers. To do this I followed a methodology very similar to how I implemented the first part of this assignment. Specifically I added an additional function called *queryLocations* to the *DataContext.js* file. This selects all of the venues currently present in the database and converts the result set to an array before calling the *processorFunc*. This time the *processorFunc* is set to a different handler that iterates over the returned data and creates markers using the Google API in *Controller.js*.

Several additional changes were made to the existing map code in the *Controller.js*. The *handle_geolocation_query* was function has been changed to remove the static map creation and instead:

- Sets up a new Google Map in the content area of the page
- Requests to load venue data from the local database
- Converts each venue to an interactive map marker

I found that this got Google maps working but encountered several additional issues which needed to be addressed. Firstly, when resizing the browser window the centre location of the map would not adjust itself accordingly. This meant that the location that the map was centred over would not adjust itself to keep the middle section always visible. I found that this was easily fixed by creating an event handler for the resize event handler available in the Google maps API (via the stack overflow question [2]).

The second issue I found after implementing this feature was that the maps would not work on a mobile device, but worked fine in a normal web browser. This is a good example of where Google Chrome's remote debugging tools came in handy during this assignment.



```
            }
            else {
                initiate_geolocation(); // Directly from the mobile brow
            }
        } else {
            initiate_geolocation(); // Directly from the browser
        }
    };

    var initiate_geolocation = function () {

        // Do we have built-in support for geolocation (either native br
        if (navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(handle_geolocation_
        }
        else {
            // We don't so let's try a polyfill
            yqlgeo.get('visitor', normalize_yql_response);
        }
    };

    var handle_errors = function (error) {  error = PositionError {}
        switch (error.code) {
            case error.PERMISSION_DENIED:
                alert("user did not share geolocation data");
                break;

            case error.POSITION_UNAVAILABLE:  error = PositionError {}
                alert("could not detect current position");
                break;

            case error.TIMEOUT:
                alert("retrieving position timed out");
                break;

            default:
                alert("unknown error");
                break;
        }
    };

    var normalize_yql_response = function (response) {
```

```
Watch
▼ error: PositionError
    code: 2
    message: "application does not have suff…
  ▶ __proto__: PositionError
Call Stack
                                    Controller.js:166
Conference.controller.handle_errors
        Paused on a JavaScript breakpoint.
Scope
▼ Local
  ▶ error: PositionError
    this: undefined
▶ Closure
▶ Global                                   Window
Breakpoints
☑ Controller.js:155
  yqlgeo.get('visitor', normalize_yql_respon…
☑ Controller.js:166
  alert("could not detect current position");
▶ DOM Breakpoints
▶ XHR Breakpoints
▶ Event Listener Breakpoints
▶ Event Listeners
```

```
Insert ID = 82                                  DataContext.js:173
Insert ID = 83                                  DataContext.js:173
Insert ID = 1                                   DataContext.js:173
Insert ID = 2                                   DataContext.js:173
Insert ID = 3                                   DataContext.js:173
Insert ID = 4                                   DataContext.js:173
```

Figure 3.2: *Screen capture showing Chrome's remote debugging tools inspecting code running on my Android phone. This shows me examining the the error variable I was receiving which stated my application had insufficient user permissions. This version used Chrome Canary so that the WebView could be debugged directly.*

Using the remote debugger tool I was able to track down the error to the fact that the application had insufficient user permissions enabled to detect the users current location. I was able to easily fix this problem once I understood it by adding the following user permission to android manifest file:

```
<uses-permission android:name=''android.permission.ACCESS_FINE_LOCATION'' />
```

4 Discussion

This implementation is obviously limited by the fact that the sessions list only returns data for the first day. The first major improvement would be to actually get this working correctly with the database for multiple days rather than being hard coded into the SQL query.

Another improvement would be to swap from the current implementation which uses the Google maps JavaScript API to use the Cordova plugin. This would bring native maps support to the project which could bring a performance improvement. However this plugin has issues with the PhoneGap/Cordova version used in this assignment so I chose to only use the JavaScript API. Upgrading to use the plugin in future work should not be a huge issue based on the current implementation.

One thing which I found noticed with this assignment was the difficulty in testing the application. Testing that the application works correctly is easy in a web context, but testing to be certain that the application works correctly is difficult when you are targeting so many devices, although the Chrome debug tools go a long way to helping. Automated unit tests for the website would not necessarily guarantee the successful deployment to a device (as in the debugging example in the previous section). I think this remains an open challenge for hybrid apps.

5 Self Evaluation

Overall I would give this assignment a 70%. I feel this grade is justified for the following reasons: I have implemented all of the requirements of the assignment brief and I feel I have clearly documented the changes made in this document and my experiences with testing/debugging the application. I also feel that while I am far from a JavaScript expert I have written code that is reasonably clean and well commented. I feel that the implementation of interactive Google maps should gain me some additional flair marks, but obviously more work could be done to the enhance the application such as the additional suggestions on the brief. Taking all of these points together I feel that a grade of 70% is not an unreasonable expectation.

References

[1] The jQuery Mobile Project. Scripting pages. `http://demos.jquerymobile.com/1.0/docs/pages/page-scripting.html`, 2011. [Online; accessed 17-Febuary-2016].

[2] StackOverflow. Question 18444161: Google maps responsive resize. `http://stackoverflow.com/questions/18444161/google-maps-responsive-resize`, 2011. [Online; accessed 24-Febuary-2014].