Scottish Weather Check

Project Evaluation

B00190919, B00240424

16/04/2014

Contents

[Scottish Weather Check 3](#_Toc386730376)

[Module Description 3](#_Toc386730377)

[Files 3](#_Toc386730378)

[cacheManifest 3](#_Toc386730379)

[index.css 3](#_Toc386730380)

[index.html 3](#_Toc386730381)

[index.js 4](#_Toc386730382)

[googlemapsapi.js 4](#_Toc386730383)

[jquery.1.11.0.js 4](#_Toc386730384)

[jquery.mobile-1.4.2.js 5](#_Toc386730385)

[jquery.mobile.structure-1.4.2.min.css 5](#_Toc386730386)

[weatherApp.css 5](#_Toc386730387)

[jquery.mobile.icons.min.css 5](#_Toc386730388)

[jgfeed.js 5](#_Toc386730389)

[Images 5](#_Toc386730390)

[Code 6](#_Toc386730391)

[Used Code 6](#_Toc386730392)

[Global Variables 6](#_Toc386730393)

[Local variables 6](#_Toc386730394)

[Functions 8](#_Toc386730395)

[Critical Appraisal 10](#_Toc386730396)

[B00240424 – HTML5, JavaScript and documentation 10](#_Toc386730397)

[B00190919 – HTML5, CSS, graphics, custom theme and documentation 12](#_Toc386730398)

# Scottish Weather Check

Web Address: <http://scotlandweather.meximas.com>

<http://scottishweather.meximas.com>

The team has developed a weather application that gets 3 day forecast information from the BBC weathers RSS feed service, the user is required to select a location from a list of predefined locations. The reason the team choose these locations was that they are all Scottish towns and cities with a population of at least 15,000, they felt this was the best way cover a large are of Scotland that has the most inhabitants. One the program gets the feeds from the BBC service it then checks through them for the relative information such as the type of weather, once the program has all the required information it then returns it in a listed format with the addition of a graphic to quickly display the type of weather for each of the three days. The team has decided to make the application a Web App, the reasoning behind this was that they wanted the application to run on as many devices as possible instead of being locked to one device if they were to do a native application. The application is expected to run on android, apple and windows mobile devices.

# Module Description

## Files

### cacheManifest

This is used to define what files should be accessed when the application is in offline mode.

### index.css

This is where the formatting for each of the elements in the application is defined.

### **index.html**

This is the html page for the application; it has been marked up to display two pages that include a home page and a weather page that shows the RSS feed information. The home page contains a header with a graphic that serves as the applications logo, included also is a dropdown menu that will change page once a selection from it has been made. The weather page has multiple div and image elements that are used to show the formatted feed data back to the user, as well as this it has a header element with a back button and a nested div element to show the selected location to the user.

### index.js

This is the applications main JavaScript file; this file is responsible for getting the selected URL and passing it through to the jgfeed file to return the feeds. If no feeds are returned the program will return the last saved feed data, once the program has the feeds they are then entered into an array to store the strings. Once it is done it then runs 3 functions that formats the data, the first function searches through the string and then writes the day of the week into its corresponding div elements. The second function will search through the strings and look for the forecast for the specified day; once it has found it the function will write the forecast in text to its corresponding div and change the source file of the days corresponding image tag. The final function looks to see if the word maximum or minimum is in the string, it then cuts the string at the start of either word and is then written into the day’s temperature div element to show the temperature for the three days.

### googlemapsapi.js

This JavaScript file is used to send the latitude and longitude values from the geo-location to google maps which will then return the users position in and address format or vice versa, the api call also be used to display a map with the users location on it.

### jquery.1.11.0.js

jQuery is an API that can be used in multiple browsers and can also be used to reduce the amount of code written in a standard JavaScript file by using new keywords and shorten code to help cut down on the amount of coded needed to be written.

### jquery.mobile-1.4.2.js

The jQuery Mobile API allows you to create applications that can work across a range of browsers on mobile devices and desktop computers; this helps to give your site and or application a clean consistent look and feel no matter what device the user is on.

### jquery.mobile.structure-1.4.2.min.css

This file allows us to use the standard method of accessing and using all of the application styling of using the standard jQuery Mobile CSS file that can give you a choice of themes, icons and other user interface abilities. The difference between the standard file and this file is that this file gives you the ability to use all of those features but allows you to easily incorporate your own custom created theme using the Themeroller from jQuery.

### weatherApp.css

This file holds the mark up to incorporate our own custom created themes that you can create using the jQuery Mobile Themeroller.

### jquery.mobile.icons.min.css

This file holds the mark up to incorporate our own custom created icon colour scheme that is also created with the jQuery Mobile Themeroller.

### jgfeed.js

This file allows you to pass a URL for an RSS feed and will then retrun the information and allow you to format and display it on your website/application.

### Images

This folder contains all of the images that are used in the application to display the different weather type and contains a default image that will be displayed if no data is received, included also is a logo for the application to be displayed on the home page and an icon that will show up while browsing the page on the browser tab and will also be used for the application icon when it has been favourite.

## Code

### Used Code

**Manifest Code:**www.html5rocks.com/en/tutorials/appcache/beginner/

**Google Maps API:** <https://developers.google.com/maps/documentation/javascript/examples/geocoding-reverse> <https://developers.google.com/maps/documentation/geocoding/#Types>

### Global Variables

**titleArray:** This array is used to store the three string elements provided using the BBC RSS feeds and jGFeed that hold all of the forecast day for a three day period.

**dayOneForecast, dayTwoForecats, dayThreeForecast:** These are three separate variables that hold each of the three array string elements, so that each one of the feed entries can be saved and called back individually.

**locationName:** This variable is used to load and store the name of the last accessed location.

**userLat/userLng:** this two variables will hold the latitude and longitude values give from the geo-location function.

### Local variables

**entry:** This is used to hold all the individual feed entries in the jGFeed API.

**title:** This is returned from the getFeed function using the jGFeed API, the title element holds all of the forecast day for one of the three days.

**dayElements:** This is an array that stores each of the three div elements that displays the day on the applications display page.

**forecastTextArray:** This array holds the three div elements that display the three days forecast in text format.

**forecastImageArray:** This array holds the three image elements that are used to display the three days forecast in graphical format.

**stringStart:** This variable is used to get the starting position of either the word maximum or minimum from each forecast.

**tempElementArray:** This array is used to hold the three div elements that display the temperature strings for each of the three days.

**tempString:** This is used to hold a string from the selected titleArrays entry starting from either the word maximum or minimum and then is displayed in the corresponding temperature display div element.

**appCache**: This variable is used to store the current version of the cache and allows the program to update and refresh the new manifest for the application.

**latlng**: This is the position from google maps that was acquired using the devices longitude and latitude values.

**location**: This holds the address returned from the google maps api and is used to get the postal\_code type which returns the postcode from the location acquired from passing the latitude and longitude values to the google maps api.

**postCode:** This variable is from google maps and inserted in-between the bbcStartString and bbcEndString to be able to access the BBC RSS information for the users location accessed through geo-location.

**bbcStringStart**: This is the first part of the url recquired to accessed the RSS information for the selected location.

**bbcStringEnd:** This is the last part of the url recquired to accessed the RSS information for the selected location.

**feedURL:** This is a single string made up of the bbcStringStar, bbcPostecodeValue and bbcStringEnd and is used as the url to be passed to the getFeed function that will get the weather forecast.

**locationName**: This is the name of the city acquired through google maps and the geo-location.

## Functions

**$(document).ready:** When the document is loaded this function checks to see if the application has an online connection, if it does it then waits for a user to click on one of the location options. When the user selects a location it sends that locations corresponding URL to the getFeed function and assigns a value to the locationName variable to be displayed showing the selected locations name. If the application does not have an online connection however it will get the last accessed information from local storage and use that as its displayed information.

**displayLocationName:** This function takes a parameter that is then used to display the selected locations name in the header of the applications displayWeather page.

**getFeed:** The getFeed function is used to get and return the different entries from the RSS feed, if no feeds exist the application will either get the last accessed information or display an error message. If the feeds exist the function loops three times adding the different entries into separate title variables, these title variables are also entered into an array to be easily accessed and then the functions to display and save the information are called.

**showFeedDays**: The function starts by inserting the div elements from the displayWeather page that will show the days into an array, it then uses a loop to go through the titleArray entries and find what day it is in the current string using indexOf. Once it has found the day that exists it inserts the text into the currently selected div element by changing its html, the loop will repeat this for all three of the entries making sure to change the div elements position to match the correct div.

**displayForecast**: The function starts by inserting the three div elements that will display the forecast for each day in text format into an array, it then does the same for the image elements that will be used to graphically display the weather. A loop is then used to cycle through each element in those arrays and loop through the entries in the titlesArray, then using an IndexOf it checks if a string matching the certain type of weather is present and if so it inserts the type of weather into the text elements and changes the source of the image elements to match the text.

**dispalyTemp**: Similar to the other functions this stores each of the three elements that will display the day’s temperatures into and array so they can be cycled through using a loop. The function then looks to see if either the word maximum or minimum exists and gets their starting point, once the starting point is found a substring is created from that point and is inserted into the currently selected div element in the array to display the temperature for that day.

**saveWeatherDetails**: This function inserts each of the titleArray entries into their own variable and then saves each one of the variables to local storage to be used if the device is offline.

**updateAppCache**: This function checks the different versions of the applications cache and if an updated version is found it will allow the user to update the applications manifest and then refresh the page to make use of the applications new version.

**getLocation**: This function will ask the user If they will allow the device to access their location using their GPS and geo-location, it will also return an error message if there is no support for geo-location from their browser.

**locationSuccess**: This function will save the users latitude and longitude values and then run the function to get their postcode.

**locationFail**: This simple gives and error message saying the program and or device could not find their location.

**codeLatLng**: This function takes the latitude and longitude recived from the device then sends them to google maps to return an address, once the address has been acquired it then looks through the result and gets a post code value. Once it has the postcode value it sets it to lower case, deletes all the white spaces and then combines it with the url for the BBC’s RSS feed and bases it to the getFeed function. The function also finds the city and sends that to the displayLocationName function.

# Critical Appraisal

## B00240424 – HTML5, JavaScript and documentation

For the project out intentions where to build a web app that displays the weather over a three day forecast for a selected number of locals in a text and graphical format, we would be doing this by using RSS Feeds provided by the BBC by using a jQuery plugin called jGFeed to return the feed information as a string. The development of the application went well we managed to achieve all the goals set out for the basic version of the application and because we finished with time to spare we managed to add in nearly all bar one of the extra’s we had planned if time allowed. These extras included an auto fill search bar to replace a drop down menu, animated graphics and a custom created theme using jQuery Mobiles themeroller.

When we were decided on the format to make the application I was wanting to create it as a Web App and that is what we finally choose to creating it as, my reasoning behind creating it as this format is that it gives the application the ability to be run on a wide range of devices as long as they have a WebKit supported browser. The drawback to this however was that by it being a Web App it could be harder to get users to the site and use the application. I did look at creating it as a native application for a specific operating system such as Android or IOS, the pros to creating it as a native application such as Android meant that we could upload it to the Google Play store and that means it would be easier for people to find and use our application. Ultimately though I felt that with the application being able to run across multiple platforms instead of locked to one feels more beneficial and useful to me and I am happy as a team we decided to create the application as a Web App.

My knowledge of JavaScript in the team was the strongest so I was given the task to create it for the application. When I was first starting the project understating how the jGFeed worked was the hardest task for me, so I started by playing around with the code to get a better understanding of how it worked and by doing this I realised that the entry.title was what stored the information we needed to display the information about the forecast, so once I had discovered this I created and array and entered them into it so I could easily access the strings individually. Once this was done I realised that all we had to do to display the information in the layout we designed was for me to use string methods to manipulate the different individual strings, I feel if spent more time looking at the jGFeed and doing some smaller applications with it I think I could have gotten a better understanding of the API and possible made the code even more efficient.

Once the basic version of the application was completed I started to implement to additional features we wrote about in the design document, the first feature we added was a search bar to make it quicker for the user to find their location. We did manage to get this implemented but it took a lot of time to find the HTML source on the jQuery Mobile page, the next feature I implemented was using the geo-location to get the weather forecast for the users location. This took a lot of work I had to search for an API to convert latitude and longitude into a post code and the only one I could find was the Google maps API, it took me around seven hours of straight coding to get the API to simply return a postcode but I was ecstatic that I managed to get it working. I do feel the code could be tidied up in the future if I were to spend more time learning about the API, while testing the application on a wide range of browsers on multiple devices and desktops I learned that there was some inconsistency’s with some of them which was quite annoying but for the majority of them the application works perfectly as intended and the ones that don’t only have the problem of no geo-location and some places take a bit of time to update their forecast data. I am not sure why exactly this happens if it is just to do with what certain devices can do or because there is a problem with jGFeed and updating the feed information, but after testing it in the browser debugging I was happy to see that even though the wrong information was returned by jGFeed our application handled the information as expected. I would be interested to use a different RSS reader API in the near future to help see if it could possibly fixes these issues.

Over all I am happy with how the application turned out and we got it running across android and IOS devices on a variant of browsers, one of the main things I learned while creating the application is that arrays in JavaScript can be used to set attributes to different elements created in the HTML document such as div and image elements. I feel if I realised this at the start I would have saved some time by not needing to copy and paste a lot of code, this could have caused errors such as having declaring the wrong element id or missing out certain parts of the coded needed. From creating this application I think I should keep working on my basic knowledge as well as more advanced techniques with HTML and JavaScript knowledge to help understand more about them both, I feel if I were to have a better understanding on how more of the JavaScript and HTML works not just my applications would benefit but creating the initial and final documentation would benefit from it.

## B00190919 – HTML5, CSS, graphics, custom theme and documentation

The final app that was developed turned out to be surprisingly similar to my first expectations of what our team was going to build. As we now have a working weather app that can show the weather for the next three days, I find myself actually using the app as a daily part of my life to find out the weather. This is because of the features of the app and how it is simple and easy to use which makes it a handy little program to have running on your mobile. I am overall very happy with the way this app turned out because at the beginning of the project, the use of the RSS feeds etc. seemed like a daunting and difficult task. On a whole I feel that I contributed as much of my skills and knowledge as I could, but I don’t think I could have completed this project without my team member as they have a greater knowledge of the programming than I do.

The tasks I done most of was creating the graphics to be used for the weather and doing the html5 for the inclusion of all the location codes and names. Along with this I completed some of the paperwork and helped out with small portions of code to be used in the main application. The majority of the programming was done by my team mate as he felt more comfortable using JavaScript and jQuery. These roles where decided early on in the project as we have known each other for some time and we know each other’s strengths and weaknesses, because of this it meant we didn’t have to spend much time at the start deciding who was doing what and when. It meant we could start almost right away.

There were only a few changes to the project that we decided on in the first few weeks of the project. Firstly is that we originally had planned to include a geo-location button so that users could check the weather at their current location. After some research of how to do this we concluded that we would leave it out for now and focus on getting the RSS feed and other features of the app working first. Another change is that we originally planned to have all the weather information display on the page but after some research and testing of the RSS feed we realized that the BBC weather feed only provided some of the information. So now the app only displays the weather and the maximum and minimum temperatures of the next three days.

A few changes that were positive ones were firstly at the beginning of the project we had a drop down menu of all the locations the weather app covered, this wasn’t ideal so we have managed to change it to a search bar that automatically fills and shows the results that the users can click on. Another positive change was to change all the weather images from single images to animations; this has given the application a better look and will make it more popular with end users. The final positive change was that I created a custom jQuery theme with custom colours to be used in the application; this also helps improve the look and feel of the app and ultimately makes it look more professional.

Despite the few negative changes and taking into account the positive ones, I am glad we chose this type of app to create and am happy with the final outcome. Things I would differently next time would be to try and split up the programming work a bit more so that I could improve and learn more, while also giving my team mate more of a help with it. Also I would do a lot more research into the type of app that is going to made to see what features would be possible in the given timetable and with the skills we have. The lessons I have learned are mostly about jQuery and other small things such as creating custom jQuery themes etc. but also that I need to be more confident in my own ability when tackling tasks that I am not sure about.