School of Computer Science
University of Lincoln
2018/19

Cross-Platform Development CMP3035M

Workshop (5) Intro to JQUERY MOBILE and basic events

Overview

Please read though this tutorial before you start!

Building on from week 1 workshop; In this workshop we will be looking at Jquery and Jquery mobile setup. We will leverage “on” and “off line” listeners in order to show a custom message if we have no internet connection and look at page events, i.e when key pages are loaded to add some simple touch and scroll events in order to show/control a third party graph library – High charts (https://www.highcharts.com/demo/dark-unica). Today we will using web browsers and PG desktop for testing. (Use Firefox for offline activity – Chrome does not support this feature)

The process is as following:

1. Download support files and set up the file PG file structure
2. Link our dependencies (jquery, jqm, styles and custom js etc)
3. Define and link our jquery mobile defaults
4. Create a multipage html document using JQM mark up (http://demos.jquerymobile.com/1.4.5/pages/)
5. Add a some links / use data attributes
6. Adding some “ready”, “deviceready” and “online/off line” listeners with functions to respond to these events.
7. Add some page event listeners
8. Add some custom event listeners for touch and scroll
9. Add a donut/pie chart to a div element when a particular page loads
10. Re animate chart on page scroll
11. Add a sound when a selector has been touched (event.preventDefault())

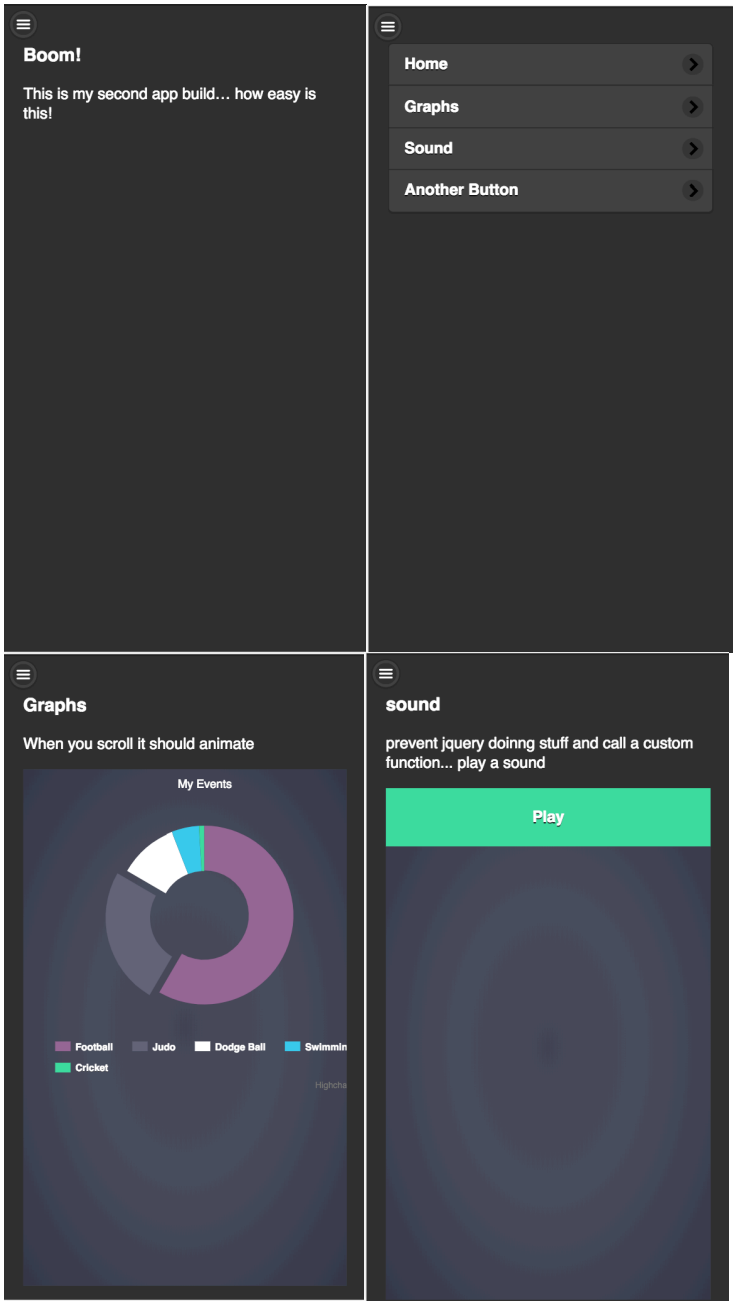
Working in the studio (optional **– I know how you like talking to each other!**)

I recommend working in **pairs*.** Find some one with a similar ability (or not) and take it turns to develop. Be proactive and resourceful. I want you to look at http://docs.phonegap.com/phonegap-build/ for reference and be creative **in your** development.

***Pair programming** is an agile software development technique in which two programmers work together at one workstation. One, the driver, writes code while the other, the observer or navigator, reviews each line of code as it is typed in. The two programmers switch roles frequently.*

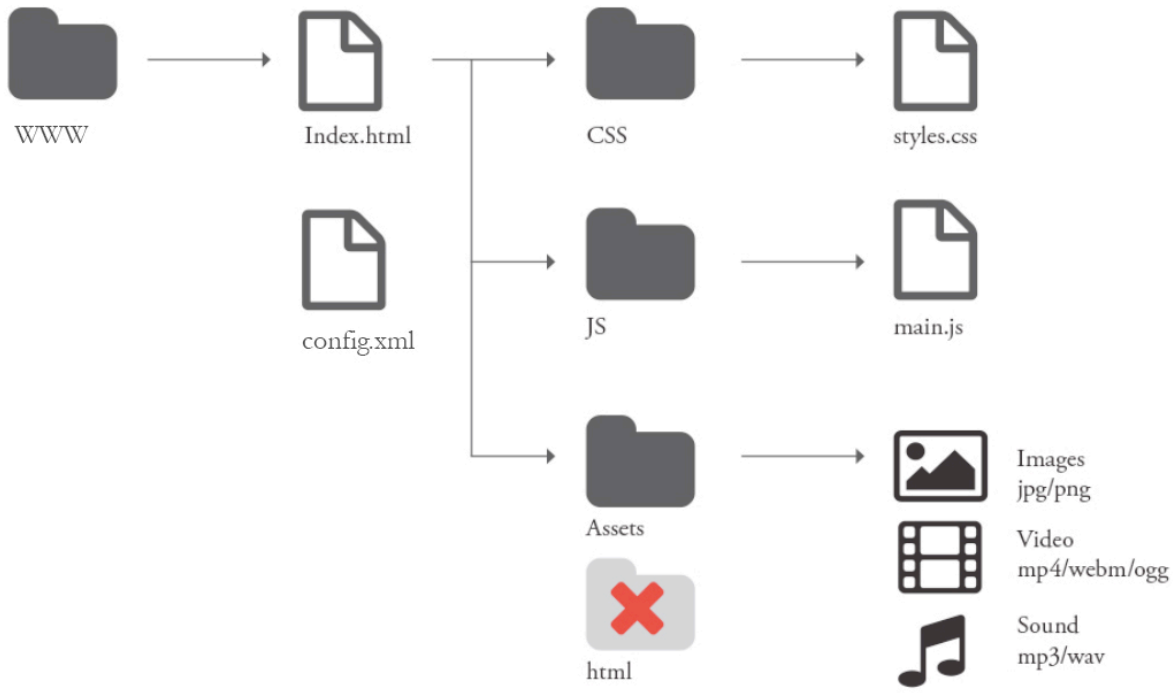
While reviewing, the observer also considers the "strategic" direction of the work, coming up with ideas for improvements and likely future problems to address. This frees the driver to focus all of their attention on the "tactical" aspects of completing the current task, using the observer as a safety net and guide.

The build



Task 1

1: Create a new folder (your_name_site_cache) with a folder structure below.



Task 2

Create your first html page, style sheet and Js dependencies

Now you have your file structure. You can now create your first page. In DW you can create a page that contains a predesigned CSS layout, or **create a completely blank page and then create a layout of your own** – do the later for now.

Go to window > workspace layout > developer

1. Select File > New.
2. In the New Document category, select the kind of page you want to create from the Document Type column. For example, select HTML to create a plain HTML page.

3. Select a document type from the DocType pop-up menu. In most cases, you can use the default selection, HTML5.
4. Select additional options depending on the type of page you want to create.
5. Save as index.html in the site cache folder you have set up. (refer to folder diagram) Notice the header and body elements have been created for you. If your not using DW then look up https://www.w3schools.com/tags/tag_doctype.asp
6. Create a new css file by going to new > css
7. Save it as custom.css in your css folder
8. Link the new css file to your webpage.
   <link rel="stylesheet" href="css/custom.css" />
9. Download jquery and add to your js folder https://jquery.com/download/
   Why do we want a physical file not just use a CDN?
10. Link jquery to your page
    <script type="text/javascript" src="js/jquery-2.2.2.min.js "></script>
11. Download Jquery mobile and the css and save in your js and css folders (use standard version for today)
    http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.js
    http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.css
12. Link jquery mobile and css to your page
13. Create a new js file by going to new > javascript
14. Link your Javascript to your page (look at lecture slides)
    <script type="text/javascript" src="js/main.js "></script>
15. Add a reference to cordova.js. Note this should **NOT** be included in your www folder. PGBuild will inject the latest build dynamically for you.
    <script type="text/javascript" src="cordova.js"></script>

Remember the order is important. It should look like following:

```
<!DOCTYPE HTML>
<html>
<head>
<title>new app</title>
<meta name="viewport" content="width=device-width, initial-scale=1">

<!--SOME APPLE SEETINGS-->
<meta name="apple-mobile-web-app-capable" content="yes">
<meta name="apple-mobile-web-app-status-bar-style" content="black-translucent">

<!--You need this although you do not need a physical reference as build will inject it in-->
<script type="text/javascript" src="cordova.js"></script>
<link rel="stylesheet" href="css/jquery.mobile-1.4.5.css" />
<link rel="stylesheet" href="css/custom.css" />
</head>

<!--notice i have added the class online which will change based on our online listeners-->
<body class="online">

```

```
<head>
        <title>myapp</title>
        <!—paths to css-->
        <link rel="stylesheet" href="css/jquery.mobile-1.4.5.css" />
        <link rel="stylesheet" href="css/custom.css" />
        <!--You need this although you do not need a physical file as build will inject it in-->
        <script type="text/javascript" src="cordova.js"></script>
</head>
```

At the bottom of the page before the closing </body> tag add all of your dependencies (js librarie and plugins)

```
80
81  <!--scripts at the bottom of the page-->
82
83  <script src="js/jquery.min.js"></script>
84
85  <!--note jqm defaults defines preferences for JQM so needs to be declared before the library-->
86  <script src="js/jqm_defaults.js"></script>
87  <script src="js/jquery.mobile-1.4.5.min.js"></script>
88
89  <!--high charts stuff with custom theme-->
90  <script src="js/highcharts.js"></script>
91  <script src="js/modules/no-data-to-display.js"></script>
92  <script src="js/themes/mypbtheme.js"></script>
93  <!--high charts stuff-->
94
95  <!--our js file goes last so we know all dependencies are loaded first-->
96  <script type="text/javascript" src="js/main.js"></script>
97  </body>
98  </html>
```

Task 3

Create a JQM Multipage Doc (HTML)

1. Create a JQM (data-role="page") with a JQM header element and JQM content element. An overview of the code is below with a code snippet to save some time.

```
<!--home-->
<div data-role="page" id="home" data-theme="b">
  <div data-role="header"> <a href="#menu" data-icon="bars" data-iconpos="notext" class="" data-theme="b" title="Menu">
  <span class="ui-btn-inner"><span class="ui-btn-text">Menu</span><span class="ui-icon ui-icon-bars ui-icon-
  shadow"> </span></span></a> </div>
  <div data-role="content">
    <h3>Boom!</h3>
    <p>This is my second app build… how easy is this!</p>
  </div>
</div>
<!--home-->
```

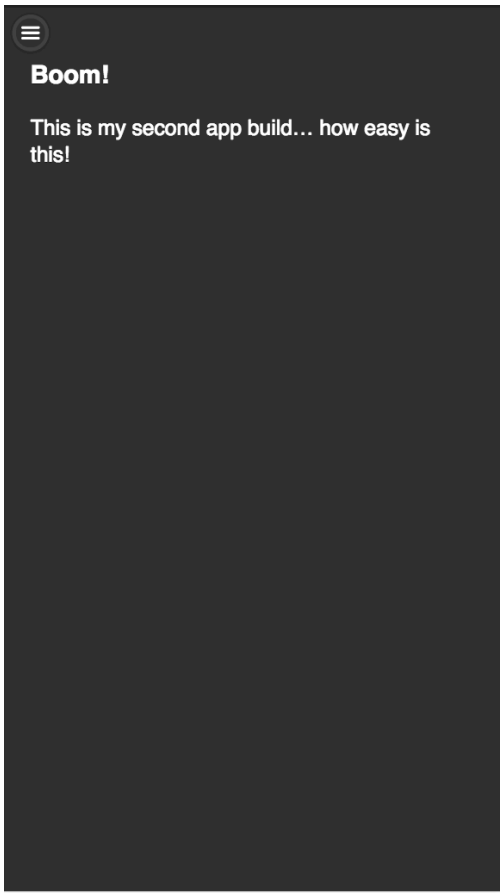
2. Paste this inside your role “header” to speed up your development. This is just a burger menu. Note that the href uses an anchor not a reference to a .html page or external site. This is because we are using multiple pages (data-role="page") in one html document. You can use separate html pages and link normally but for this workshop use a multipage page set up.

Menu

Note the new html5 data attributes being leveraged by JQM. data-theme a is white, data-theme B is Black. Jquery mobile does the styling if we want to use it.

3. **Save this as index.html in the www folder.**

4. View your page in the browser for some html debugging. Use developer tools and view on a device. Note there will be some errors in the console as there is no Cordava – but this is good for quick CSS and basic web functionality tests instead of deploying to a phone.



Task 4

Add device listeners (JS)

1. Open your **main.js file**. We re going to listen for the browsers document ready event and Phonegaps “devicready” event and call an innit function this means we can simulate the “deviceready” event in the browser. Device ready only gets called in a Phonegap app on a device.

**The use strict will hide a few errors in DW. This just means that we declare all variables with var instead of just assigning them at runtime.*

```
"use strict";

var audioElement;

function onDeviceReady() {
    console.log("device ready");
    innit();
}

$(document).ready(function () {
    console.log("ready!");
    innit();
});


function innit() {

}

```

2. **Add "internet connection" listeners (JS)**

So far so good – lets add some event listeners for checking if the user is on or off line (we will add these functions after). We also need to check if the device is online as soon as the page is loaded otherwise the event listener will not fire as the event listeners are waiting for a **connection change**. This uses **window.navigator.onLine** method which returns true or false – on or off line;
Notice here we are using jquery to add a CSS class to the body. If you look in the css is provided you will see a reference to this that sets an item to display none.

```
body.online #offline{
display:none;
}
```

The point here is to have a hidden div that only shows if there is **no** Internet connection. What we do with this item will become apparent next.

```
function innit() {

    document.addEventListener("online", onOnline, false);
    document.addEventListener("offline", onOffline, false);

    //check for internet as soon as device is ready
    if (window.navigator.onLine) {

        //this adds a class to the body tag which CSS will style (show)
        $('body').addClass('online');

    } else {
        ////console.log('offline');

        //this adds a class to the body tag which CSS will style (hide)
        $('body').addClass('offline');

    }
}
```

3. We still need to add functions that fire on the connection change which does exactly the same as above but the innit function is only fired once.

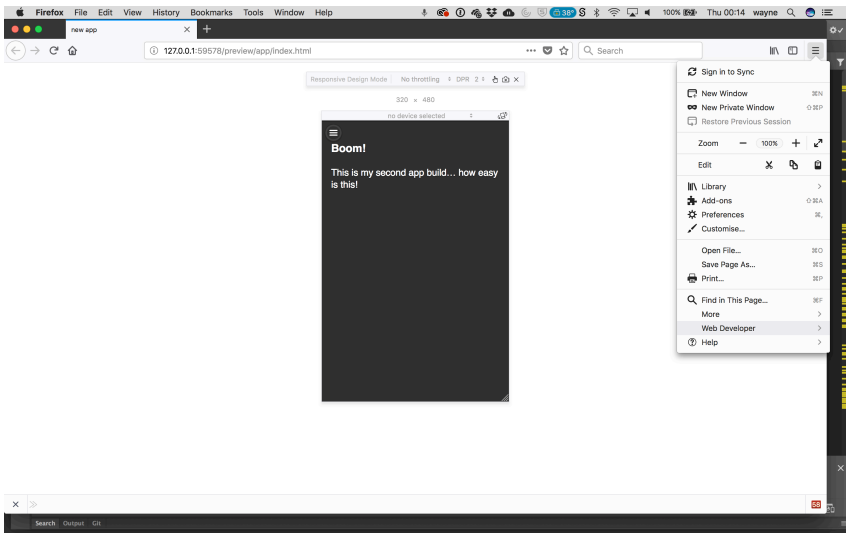
```
35
36  // handle online and offline intermittent connectivity
37  function onOffline() {
38      // Handle the offline event
39      // adding and removing classes - much easier than manually restyling in js
40      $('body').removeClass('online');
41      $('body').addClass('offline');
42  }
43
44  function onOnline() {
45      // Handle the online event
46      $('body').addClass('online');
47      $('body').removeClass('offline');
48
49  }
```

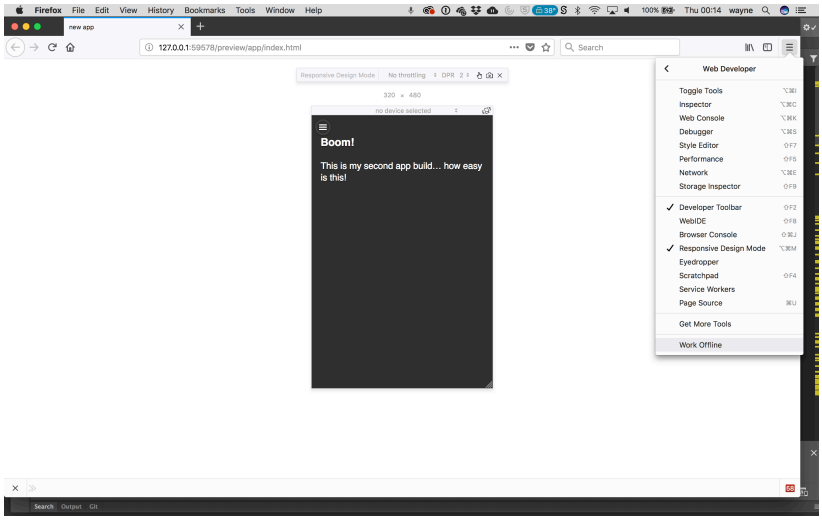
4. Now we need to add the **HTML** mark up that displays that message. The following mark up goes into you **index page**. Notice that this has no JQM data-role attribute. We want to keep this accessible at all times WITH NO ENHANCEMENTS. Remember we will only see this when we go off line.

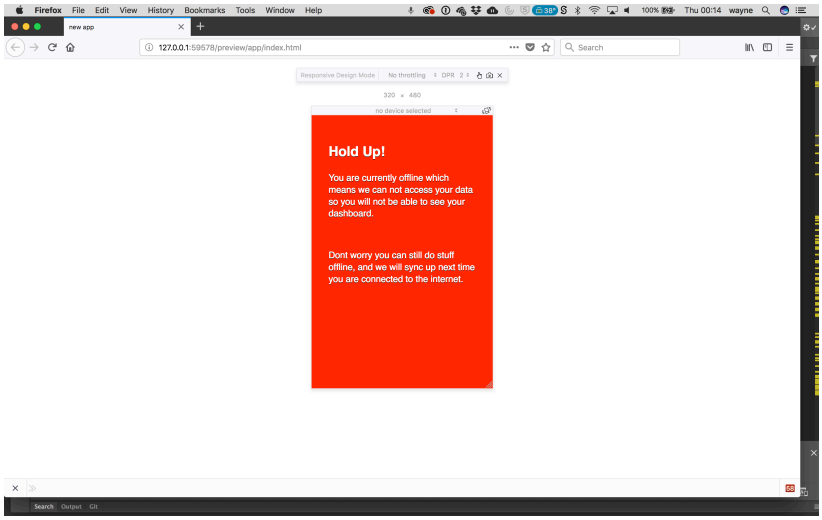
```
20
21  <!--note this is outside of the page role so will never get removed and is always accessible-->
22  <div id="offline" class="fixed">
23    <h2>Hold Up!</h2>
24    <p>You are currently offline which means we can not access your data so you will not be able to see your dashboard.
      </p>
25    <br>
26    <p>Dont worry you can still do stuff offline, and we will sync up next time you are connected to the internet.</p>
27  </div>
28  <!--end offline display-->
29
```

5. Check we are off line in the browser (Quick debugging)
   To check the code is working we need to open our index.html in Firefox. Chrome dev tools does not fire any events when in offline mode but FF does.

   In Firefox go to menu>web dev tools >Work offline. Your body class should change and therefore your CSS styles will kick in.







Boom – lets do some extra pages and event listeners

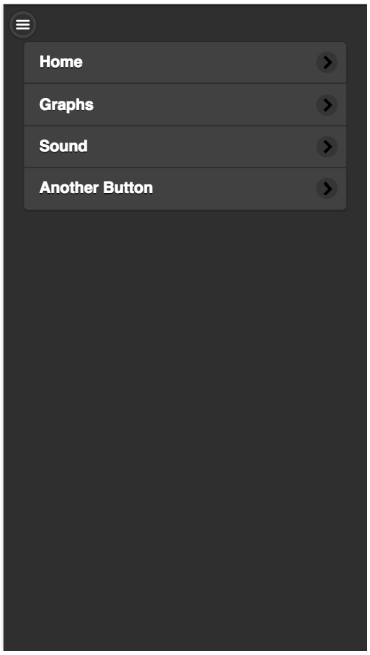
Task 5

Here we are going to create a graph Page to display some data (I have created this for you). Learning high charts is not the point – they just look nice. This is about initiating some JS when a specific page is loaded.

In your Index.html we need to add another 2 JQM pages. A div with the JQM data-role-"page" attribute. This tells JQM to hide it until we navigate to it. Here we are adding a menu page for our interface and another page which has 2 containers to load out graphs in.

1. Create the menu page – this is linked in the header code you copied previously. It links to href=”#menu”. Note the ID is menu

```
65
66  <!--menu-->
67 ▼ <div data-role="page" id="menu" data-theme="b">
68    <div data-role="header"> <a href="#menu" data-icon="bars" data-iconpos="notext" class="" data-theme="b" title="Menu">
      <span class="ui-btn-inner"><span class="ui-btn-text">Menu</span><span class="ui-icon ui-icon-bars ui-icon-
      shadow"> </span></span></a> </div>
69 ▼  <div data-role="content">
70 ▼    <ul data-role="listview" data-inset="true" class="ui-alt-icon" data-theme="b" >
71        <li><a href="#home">Home</a></li>
72        <li><a href="#graph">Graphs</a></li>
73        <li><a href="#sound">Sound</a></li>
74        <li><a href="#">Another Button</a></li>
75      </ul>
76    </div>
77  </div>
78  <!--menu-->
79
```



Below is a code snippit for an unordered list – which is our navigation. Pate this inside the content div to save a little time. Note each link has an anchor on the href.

```
<ul data-role="listview" data-inset="true" class="ui-alt-icon" data-theme="b">
   <li><a href="#home">Home</a></li>
   <li><a href="#graph">Graphs</a></li>
   <li><a href="#sound">Sound</a></li>
   <li><a href="#">Another Button</a></li>
</ul>
```

2. **Save and preview in a browser.** Clicking on the burger menu will animate in the menu page. The animation settings are set in JQM defaults but can be defined in the a link if you want it to be different for example
I'll pop

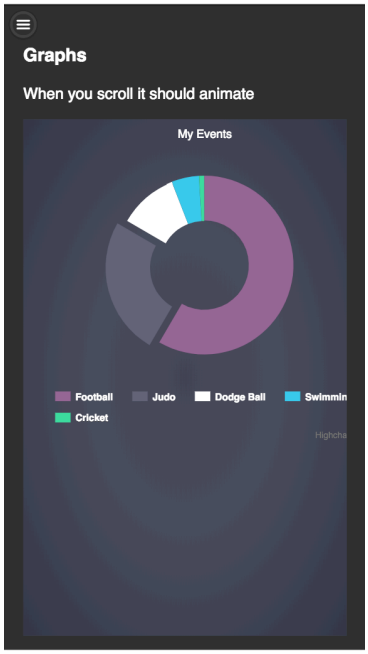
3. Add a generic and specific page load event which means we can call a custom function to generate our graph. Uncomment the alerts and you will see the order events get fired. This is important depending on what you want to do and when you want to do it.

Generic events below (JS)

```
54 $(document).on("pagebeforeshow", function () {
55     // When entering pagetwo
56     //alert("page is about to be shown");
57
58 });
59
60 $(document).on("pagecontainershow", function () {
61     // When entering pagetwo
62
63 });
64
65 $(document).on("pagecontainerload", function (event, data) {
66     //alert("pageload event fired!");
67 });
68
69 $(document).on('pagecreate', '#menu', function () {
70     console.log("pagecreate menu");
71
72 });
```

Add the mark up to the page. Notice how most of it is similar however I have added 2 new divs with id donut and line. These are basically holders for graphs which are generated by JS when the graph page is loaded.

```
40
41  <!--graph-->
42 <div data-role="page" id="graph" data-theme="b">
43   <div data-role="header"> <a href="#menu" data-icon="bars" data-iconpos="notext" class="" data-theme="b" title="Menu">
     <span class="ui-btn-inner"><span class="ui-btn-text">Menu</span><span class="ui-icon ui-icon-bars ui-icon-
     shadow"> </span></span></a> </div>
44   <div data-role="content">
45     <h3>Graphs</h3>
46     <p>When you scroll it should animate</p>
47     <div id="donut" class="graph"></div>
48     <div id="line" class="graph"></div>
49   </div>
50 </div>
51  <!--graph-->
52
```



Specific page event

```
92 ▼ $(document).on('pagecreate', '#graph', function () {
93       console.log("pagecreate graphs");
94
95       //call chart functions
96       create_pie_chart();
97       create_line_chart();
98
99 ▼     $(document).on("scrollstop", function (event) {
100          //gets y position of selector to see how far it has been scrolled from the top
101          var piechart = $('#donut').offset().top;
102
103          //gets y position of selector to see how far it has been scrolled from the top
104          var linechart = $('#line').offset().top;
105
106          //gets y position of window to calculate if its in the viewport, if it is then animate it
107          var topOfWindow = $(window).scrollTop();
108          var h = $(window).height();
109
110          console.log(piechart + ' ' + topOfWindow);
111 ▼        if (piechart < topOfWindow + h) {
112              create_pie_chart();
113          }
114 ▼        if (linechart < topOfWindow + h) {
115              create_line_chart();
116          }
117      });
118
```

Specific event for graph page with a scroll event listener which activates the graph animation on scroll stop. The functions have been provided for you.

Additional

Extra – so this is quite loose.

1. Create a new JQM page with a button in it

2. Add another page specific listener for a sound page and then add an ontouch event to play a sound.

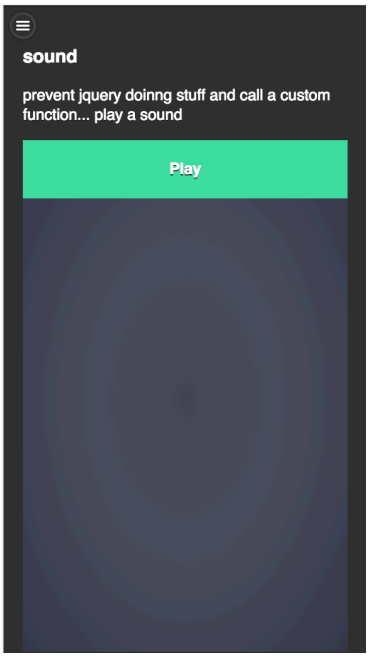
Markup

```
53
54  <!--sound-->
55 ▼ <div data-role="page" id="sound" data-theme="b">
56    <div data-role="header"> <a href="#menu" data-icon="bars" data-iconpos="notext" class="" data-theme="b" title="Menu">
      <span class="ui-btn-inner"><span class="ui-btn-text">Menu</span><span class="ui-icon ui-icon-bars ui-icon-
      shadow"> </span></span></a> </div>
57 ▼   <div data-role="content">
58      <h3>sound</h3>
59      <p>prevent jquery doing stuff and call a custom function... play a sound</p>
60      <div class="graph"> <a id="soundbtn">Play</a> </div>
61    </div>
62  </div>
63  <!--sound-->
64
```

JS

```
75
76 ▼ $(document).on('pagecreate', '#sound', function () {
77       console.log("pagecreate sound");
78
79 ▼     $('#soundbtn').on('click', function (event) {
80               console.log("play");
81           event.preventDefault();
82           audioElement = document.createElement('audio');
83           audioElement.setAttribute('src', 'assets/sound/beep.mp3');
84
85           audioElement.play();
86           navigator.vibrate(1000);
87       });
88
89  });
90
```

Make sure **var audioElement** is declared on top of the page to give global scope.



Remember

Always SAVE your work regularly on your own University storage. Backed up by ICT).

Always BACKUP your work regularly, into more than one extra storage (e.g. USB drive, DropBox, Google Drive,…….).

The lab machines can be wiped and/or re-ghosted at any time, without warning. Nothing should be stored in these machines.