PROGRAMMING FOR MOBILE DEVICES

Programming Mobile Devices
Lecture 3: jQuery Mobile

A McMonnies
School of Computing

13/02/2014 1

Web Apps for Mobile

- We can use HTML+CSS+Javascript to create a web-app
- We can also (with a fair bit of work) customize a standard web-app to work on a mobile device
  - See ***Building iPhone Apps with HTML, CSS and Javascript, Jonathan Stark, O'Reilly pub***
    - This would lead to apps that work ***only*** on an iPhone (not a good development principle)
    - For e.g. an Android device, different style sheets/Javascript etc. are needed
- The ideal is for someone to produce a cross-platform Mobile-Apps framework
  - Not as difficult as it might seem – iPhone, Android and (to an extent) Nokia use Web-kit as a browser core, so are compatible *to a degree*
- jQuery Mobile goes beyond this
  - Web-apps built with this will ***adapt*** to different browsers and devices
    - See JQM Mobile Graded Browser Support (http://jquerymobile.com/gbs/)
- Result – we can build a generic mobile app that will operate the same way in a range of current devices

13/02/2014 2

jQuery Mobile app-structure

- jQuery Mobile (jQM) is built on top of jQuery
  - Same basis – selectors for DOM items, but adds CSS for styling (also animations/page transitions)
- Much of JQM works automatically
  - Include the libraries (style-sheets *and* Javascript) and a page will automatically style to suit a mobile device – AJAX is used for this
  - Use some standard tags (e.g. , <li> etc.) and get styles well suited to a mobile device
- JQM's CSS support includes some very neat tricks
  - A standard HTML page gets re-styled on the fly to appear as a web-app on the device – AJAX functions replace existing content with specific styling
  - Because of this, different styling can be applied to different devices
  - Elements and styles are added to pages automatically, so that what is rendered in the browser is very different from what was served from the website
- Many of the standard iPhone and Android native application features (e.g. slick page transitions, list-based pages) can be replicated in a HTML/CSS/JS web-app with much less effort and cross-platform capability thrown in for free

13/02/2014 3

Organization of a simple JQM app

- Each JQM page (in HTML) can contain multiple 'pages'
  - A (block division) element can be used to create a separate page…

```
<div data-role="page">
        Page content here..
</div>
```

  - Separate elements can be used to organize the page into header, content and footer sections…

```
<div data-role="page" id="one">
        <div data-role="header">
                …
        </div>
        <div data-role="content">
                <!– include page links-->
        </div>
        <div data-role="footer">
                …
        </div>
</div>
```

Single Page Structure

```
<html>
<head>
        <title>Page Title</title>
        <link … (to CSS style sheets etc./>
        <script … (urls of jQuery/JQM)> </
script>
</head>
<body>
        <div data-role="page" id="one">
                <div data-role="header">
                        …
                </div>
                <div data-role="content">
                        …
                </div>
                <div data-role="footer">
                        …
                </div>
        </div>
        <div data-role="page" id="two">
                … etc …
        </div>
        <!– more page definitions -->
</body>
</html>
```

Multi-Page Outline

13/02/2014 4

Links between pages

- elements are used extensively in JQM, along with the data-* attribute
  - See http://ejohn.org/blog/html-5-data-attributes/ for an explanation
- Since a is probably a significant part of a page-structure, significant elements should be given a unique identifier – the HTML id attribute does this…

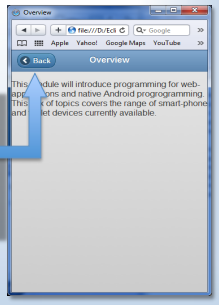
```
<div data-role="page" id="main">
        Page content here..
</div>
```

- An anchor (<a>) element can be defined as a button to form links between pages…

```
<div data-role="page" id="sub">
        <a href="#main" data-role="button">Return to main</a>
</div>
```

- A 'back' button is just an anchor/button with data-rel="back"

```
<div data-role="page" id="sub" >
        <div data-role="header">
                <a href="#" data-role="button" data-rel="back">Back</a>
        </div>
        <p>No need for a return to main button here.</p>
</div>
```



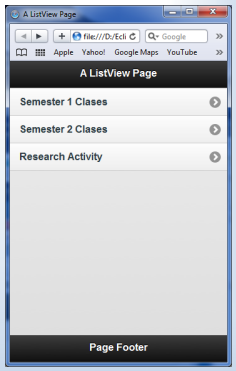
13/02/2014

Listview pages

- Used for either links to multiple pages, or for displaying a list of data-items
  - e.g. mail messages, appointments, photo thumbnails

```
<div data-role="page" data-role="listview" id="main">
        <div data-role="header">
                <h1>A ListView Page</h1>
        </div>
        <div data-role="content">
                <ul data-role="listview">
                        <li><a href="#sem1">Semester 1 Classes</a></li>
                        <li><a href="#sem2">Semester 2 Classes</a></li>
                        <li><a href="research-activity.html" rel="external">
                                Research Activity</a></li>
                </ul>
        </div>
        <div data-role="footer" data-position="fixed">
                <h4>Page Footer</h4>
        </div>
</div>
```

Note – this is a link to a separate HTML file (no #). The others link to page s in the same file.



13/02/2014

Manipulating Listviews

- Often a ***listview*** is used to display content that you want the user to be able to manipulate e.g.
  - add an item
  - remove an item
  - edit an item
- In this situation, Javascript (& jQuery) code are used to provide the facility

See List Demos on Moodle (Tools and other goodies) for a complete set of functions for manipulating items in lists.

```
function addItem(){
  var toAdd;
  popupPrompt("What is your name", function() {
    if( toAdd !== "") {
      addNewItem(toAdd);
    } else {
      popupAlert("Not added");
    }
  });
}
function addNewItem(itemText){
      var listitem = newListItem(itemText);
      var list = $("#list");      // ID of UL or OL element
      list.append(listitem);
      list.listview("refresh");
}
function newListItem(itemText){
      var item = document.createElement('li');
      item.innerHTML = itemText;
      return item;
}
```

Note – These are not standard. See "Dialog Boxes" later

jQM Page Transitions

- JQM provides a set of CSS-based transition effects that can be used to make page changes more visually appealing
  - A new page can appear with any of the following effects:
    - slide (left to right)
    - slideup
    - slidedown
    - pop
    - fade
    - Flip
    - flow – (shrink→slide→grow)
- The transition is defined at the link end - i.e. it is defined in an <a> (anchor) tag that can be applied to a list item, button or on its own. e.g.
  - Go to Index
  - Intro to Programming for Mobile Devices
*
  - <button>View Results</button>
- This may seem trivial, but to end-users the polish added by page transitions tends to reinforce the distinction that this is not a simple website.

13/02/2014 8

Dialog Boxes

- In User-Interface terms, a dialog-box (or simply 'dialog') is a way to ***pause an application*** until the user has completed some task
  - Reading a message
  - Providing some required data
  - Confirming an operation
- A dialog-box is a "modal" component (i.e. it has a 'mode')
  - When the user is presented with a dialog box, the message is "***do this, before anything else will happen***"
- jQM can create a dialog-styled page very simply:
  - Print Page
  - Note that it is the hyper-link end that says that the link is to a dialog. The actual dialog is just a as usual, although data-role='dialog' is supported too
  - The page has a different appearance (it is inset, to appear as a small window floating above a blank screen set to the calling page's theme
  - A small close-button (a cross) is added to the left of the page header
  - The user either performs the required action (e.g. selecting an item from a list) or presses the close button to cancel
  - You can add OK/Cancel buttons if you like – just need to provide the right <a> tags
- The current version of jQM also provides "pop-ups" : similar to dialogs, but prettier
  - See jQuery Moble Dialogs on Moodle→Tools and other goodies

13/02/2014 9

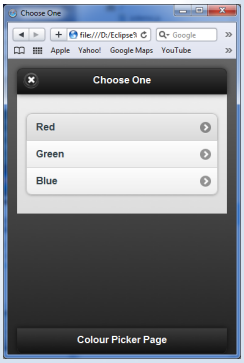
Dialog Box Example

- Call on a dialog like...

```
<button><a href="#colour_dialog" data-rel="dialog">Select a colour</a></button>
```

- The dialog can be defined as...

```
<div data-role="page" data-add-back-button="true" id="colour_dialog">
  <div data-role="header">
    <h1>Choose One</h1>
  </div>
  <div data-role="content">
    <ul data-role="listview" data-inset=="true">
      <li id="red"><a href="#sem1" data-rel="back">Red</a></li>
        <li id="blue"><a href="#sem1" data-rel="back">Green</a></li>
      <li id="green"><a href="#sem1" data-rel="back">Blue</a></li>
    </ul>
  </div>
  <div data-role="footer" data-position="fixed">
    <h4>Colour Picker Page</h4>
  </div>
</div>
```



- Note that we still need some JS code to process the response.

13/02/2014 10

Processing a dialog response

- As usual, we can use $(document).ready() to set up event handlers
  - In this case, an alert box, but we could set a ***colour*** variable, change the colour of a screen element or anything else
- Note that setting up three individual event-handlers could get tedious
  - A better approach is to make all of the colour list items of the same class (add an attribute 'class="col_pick"')
  - We can now use a much more concise selector to do the same job
  - Note
    - Selector starts with a dot (class) instead of a **#** (id)
    - We can use ***this.id*** to refer to the id of the element that was clicked on

```
$(document).ready(function(){
  $("#red").bind('click', function(){
    alert("Picked Red!");
  });
  $("#blue").bind('click', function(){
    alert("Picked Blue!");
  });
  $("#green").bind('click', function(){
    alert("Picked Green!");
  });
});
```

```
$(document).ready(function(){
  $(".col_pick").bind('click', function(){
    alert("Picked "+this.id);
  });
});
```

13/02/2014 11

Pre-Defined Dialogs

- A better approach is to define some dialog box types in code (a mix of HTML and JS required). Useful ones would be the jQM equivalents of
  - Alert (Show a message – press OK to cancel) – e.g. popupAlert("Surprise!");
  - Confirm (Press Yes or No/Ok Or Cancel to continue) – e.g. result = popupConfirm("Really?", yesFunc, noFunc);
  - Prompt (User enters some text or data and Ok or Cancel) – e.g. popupPrompt("Enter your name", okFunc, cancelFunc);
- See Dialogs.zip in the Tools and Other Goodies folder on Moodle
  - Note that the code in Dialogs.zip now uses the jQuery Mobile "popup" format, which gives a more polished look to dialog boxes.

13/02/2014 12

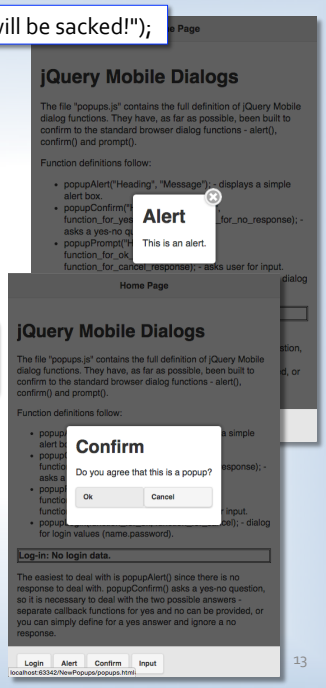
```
popupAlert("Error", "Some programmer will be sacked!");
```

Call Format:
```
popupAlert(title,
message);
```

```
popupConfirm("What Day", "Is it Tuesday?", function () {
  // Do this if OK...
  $("#result").text("Answer was OK");
}, function () {
  // Do this if cancel...
  $("#result").text("Answer was Cancel");
});
```

Call Format:
```
popupConfirm(Title, Question, functionIfYes,
                                functionIfNo);
```

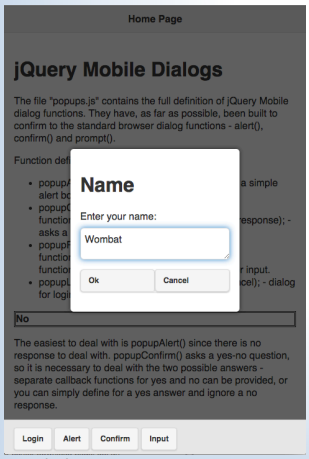
- Note the use of event handling functions to process the responses to popupConfirm()
  - This is the way to handle user-input in Javascript
  - popupConfirm() can't just return true or false, because that would mean the function call had to 'block' processing
  - The functions don't have to be anonymous – you could use named functions



13/02/2014 13

```
popupPrompt("Name", "What is your name?", function () {
    // Do this if OK was pressed...
    $("#result").text(getPromptValue());
}, function () {
    // Do this if cancel was pressed (could be omitted)...
    $("#result").text("No name was entered.");
});
```

Call Format:
```
popupPrompt(Title, Question, functionIfOk,
                              functionIfCancel);
```



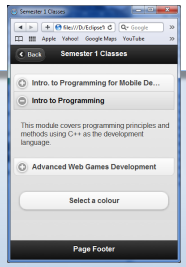
- This also needs to be 'non-blocking'
  - In this case, it is easier to provide a function that retrieves the result from the dialog box
  - popupPrompt() asks the question
  - getPromptValue() gets the response
  - Usually, call getPromptValue() inside the response function

13/02/2014 14

Collapsible Content

- Mobile devices mostly have one thing in common – a small screen size
  - Because of this, we need to pick up some tricks to minimize the amount of space content occupies
  - Using page/divs id is one good way to do this
    - Break content into chunks, each fitting on a page
- An alternative is to use the 'collapsible' attribute on a div
  - Make a collapsible, and a header within it (style H1-H6) will gain an icon to indicate it can collapse any content
  - All content after the header will be collapsed (i.e. invisible and taking up no screen space)
  - Click on the header to access the hidden elements
- A collapsible set can be used to organize a number of collapsible elements so that only one is 'open' at a time
  - Click on one to open it, others are collapsed

```
<div data-role="collapsible-set">
  <div data-role="collapsible" data-collapsed="true">
    <h3>Intro. to Programming for Mobile Devices</h3>
    <p>This module follows on from COM07027,
        Introduction to programming, and covers the
        development of web-apps for mobile devices in the
        first part, and native Android development in the
        second part.</p>
  </div>
  <div data-role="collapsible" data-collapsed="true">
    <h3>Intro to Programming</h3>
    <p>This module covers programming principles and
        methods using C++ as the development
        language.</p>
  </div>
</div>
```



13/02/2014

jQuery Mobile Development Principles

- First, ***design*** your application
  - What screens/pages are needed (pencil and paper is very good for this)
  - What are the best interaction methods for a specific requirement
    - E.g. How to get text/numbers/choices from the user
    - How to provide feedback
    - What happens while a lengthy process goes on (users hate this)
- Next, use a ***decent*** programmers' text editor
  - Notepad++, HTML-Kit etc. are good on a PC
  - TextWrangler (or the purchased version – TextMate) are good on a Mac
  - Loads of good Linux editors also available
  - WebStorm is very highly rated
    - I've found this to be the best working environment, and I'll provide some keyboard macros to speed up coding
    - Live Templates (settings.jar from Moodle) simplify jQM mark-up
    - Web Server built-in – this will save lots of problems when you get to use AJAX
- Start from a page template
  - There is no point re-developing a page structure from scratch every time
  - jQM Live templates – jqm+Tab (inserts links to jQuery Code), jqmpage+Tab (inserts page mark-up)
- Test continuously
  - Javascript has a bad tendency to just do nothing if it meets bad code
  - Test EVERY TIME you change some code
  - Write a test for every function you write
    - Sometimes loads of tests are needed
  - Make it easy to run a test
    - Use Jasmine test suites – these are now incorporated into WebStorm templates on Moodle

13/02/2014 16

jQuery Mobile Development Principles II

- Stage-by-stage

1. Design all of the pages of your app using pencil+paper, or drawing application
2. Mock-up the user interface using only HTML and jQuery Mobile. No code at this point. Make sure to give every significant element a unique ID (all list items, buttons, text boxes, every div that you may need to get data from or insert data into etc.)
3. Define the core data needs
   - e.g. an array of currency name+value pairs, a data-table of customer info etc.
   - Place this at the top of a JS file
   - Test you can access it
4. Add functions to manipulate the data as required by the application
   1. At this stage, ignore the user-interface – you're testing functionality
   2. Provide at least one test ***every time*** you write a function or method– work on the function until it passes (all) the test(s)
   3. Don't delete these tests – define them in js files in a separate folder (e.g. spec/)
5. Once all of the application functionality is in place, 'wire' up the user-interface elements. Use the ***$(document).ready()*** method (or pagecreate() if using Ajax – later) to bind event handlers to controls – lots of options in Lab 4
6. Test the user-interface to ensure the application behaves as required

13/02/2014 17

jQuery Mobile Info

- For now, the best source of information is the jQuery mobile home page
  - http://demos.jquerymobile.com/1.4.5/ (latest stable version, January 2015)
  - This is frequently updated
  - There are now loads of books
  - jQuery Mobile by Jon Reid is the biggest load of ~~bu****it~~ crap I've spent money on in recent years – don't go there
  - jQuery Mobile First Look by Giulio Bai was ok, but is well out of date
  - Pre jQuery Mobile by Broulik is the best I've found so far (but far from perfect). Ok for £24.95
  - jQuery Mobile Develop and Design is not bad, and very pretty at £23.50 or so
- There are several "available for pre-order" books on Amazon
- For a specific problem, do a Google search
  - E.g. "jQuery mobile dialog return" got me to a couple of pages that showed how to use the values entered in a dialog box
  - There is a thriving jQuery Mobile development community out there, and many of them are happy to provide advice and assistance
  - Remember, however, this is the web, and just as many people will tell you rubbish – learn to discriminate the idiots from the knowledgeable people

13/02/2014 18

Lab Work

- Last week's lab was a short intro. to jQuery Mobile web-apps
- This week, we'll build a more realistic jQuery mobile app that uses an online service
  - Currency conversion
    - done very differently from the version in the HTML5 module
- Aim to have a small application working (at least in a browser) by the end of today
- Next week – deeper into Javascript and OOP
  - Adding functionality to mobile apps
  - Adding data, structure and events

13/02/2014 19