Birkbeck
UNIVERSITY OF LONDON

jQuery Mobile Navigation

Mobile Application Development

Session 4

When the requested page loads, the jQuery Mobile parses the document for an element with the `data-role="page"` attribute and inserts that code

jQuery Mobile Navigation

- jQuery Mobile includes a unique navigation system which loads pages into the DOM via AJAX, enhances the new content, then display pages with animated page transitions.
- The navigation system automatically *hijacks* the default behaviour of standard links and form submissions, and routes them as an AJAX requests to the server.
- Using AJAX means that new pages can be loaded without full page reloads, and page transitions are possible.

jQuery Mobile Navigation

- When a new page is requested, jQuery Mobile parses the page and inserts that code into the DOM created when the first (.html) page was loaded.
- Then, any *widgets* in the incoming page are enhanced to apply all the required jQM styles and behavior.
- Other parts of the incoming page (e.g. scripts, stylesheets, etc.) will not be included. For this reason all style and script information should be included in the head of *every* (HTML) page. *
- Once the requested page is in the DOM and enhanced, the page transition is applied and the new page is animated into view.

Hash Change Tracking

- All navigation in jQuery Mobile is based on changes to the *location.hash* property.
- Hash changes are created whenever a link is clicked.
- Hash values created by jQuery Mobile are rendered as full paths relative to the URL of the first (.html) page.
- The hash is always maintained as a valid URL, so any page in jQuery Mobile can be bookmarked or referenced in a link.

```
http://www.jqm.com/transitions.html
http://www.jqm.com/transitions.html#second
```

- Hash changes that occur independently of a click, e.g. clicking he back button, are dealt with through the *hashchange* event, which is bound to the window object.

Mobile App Navigation Design

- In jQuery Mobile, the content section rather than the header is used as the container for primary navigation elements.
- The standard approach is to use a *listview* to organise navigation elements; however, a *dashboard* approach using icons can also be employed where navigation options are limited in number.

http://mobile.united.com/
http://m.domekoto.com/

jQuery Mobile Listview

- A *listview* is one of many jQuery Mobile *widgets*.
- In essence, they are a simple unordered list with a set of jQuery Mobile styles and methods associated with it.
- To create a listview, we simply need to add a *data-role* of *listview* to an existing unordered list.
- To make the list items navigable we just add standard links.

```
<ul data-role="listview">
   <li><a href="#second">Secondary Page 1</a></li>
   <li><a href="#third">Secondary Page 2</a></li>
   <li><a href="#fourth">Secondary Page 3</a></li>
   <li><a href="#fifth">Secondary Page 4</a></li>
   <li><a href="#sixth">Secondary Page 5</a></li>
</ul>
```

- Example

Enhancing jQuery Mobile Listview

- List views can be enhanced in several different ways:
  - Insetting the list
  - Dividing the list
  - Auto-dividing the list
  - Making the listview searchable
  - Using thumbnails and icons

Inset Listview

- Insetting a listview wraps and insets it within the available screen real estate.
- In addition, it adds rounded corners to the list border.
- To create an inset list we simple add *data-inset="true"* to the <ul> element as a property and value.
- Example

Listview with Dividers

- Where lists are long with lots of items, jQM helps us to make them easier to traverse by allowing us to divide them into distinct categories.
- To do this we simply need to add appropriate *list dividers* to the listview.

```
<li data-role="list-divider">South America</li>
```

- Example

Listview with Auto-dividers

- We can also let jQuery Mobile auto divide a listview alphabetically, using the *data-autodividers* property in the <ul> element.

```
<ul data-role="listview" data-inset="true" →
data-autodividers="true">
```

- Example

Searchable Lists

- The power of jQM is demonstrated by its searchable listview feature.
- Here, jQM allows us to add a search field above the list. Entering character values in the search field will automatically narrow the choices in the listview.
- To make a listview searchable, we simply add a *data-filter* property and set it to *true*.
- We can also add placeholder text to the search field using the data-filter-placeholder property, and specifying a text value for the default text.

```
<ul data-role="listview" data-inset="true" →
data-filter="true" data-filter-placeholder="Choose a Country">
```

- Example

Toolbars

- We can think of jQM toolbars as similar to common links bars in desktop apps.
- They should contain menu choices such as, *contact*, *about*, *login*, *desktop site*, etc.
- Toolbars can be positioned in the header or footer of a jQM app.
- There are pros and cons to both positions.
  - In the header, the toolbar buttons are wrapped around, and compete with, the page <h1> heading.
  - In the footer, the toolbar may be competing with the mobile browser native toolbar.
- In general, it is recommended that you use the header for global navigation elements like *home* or *back*, or for functional elements like *save* and *cancel*, and use the footer for common elements, such as *contact*, *about*, login, etc.

Toolbars

- We can create toolbars by simply adding buttons to the header or footer of a page.
- In the header buttons are automatically positioned either side of the <h1> heading. In the footer, they are automatically displayed inline.
- In footer menus, we can make the buttons look more like menu items by applying the jQM specific class, *ui-bar*, to the footer .

```
<div data-role="footer" class="ui-bar">
  <a href="contact.html">Contact</a>
  <a href="about.html">About</a>
  <a href="login.html">Login</a>
  <a href="register.html">Register</a>
</div>
```

- Example

Fixed Toolbars

- In the previous example, you will notice that the footer menu is always positioned at the bottom of the page. This means on smaller screens, or on pages where there is a lot of content, the menu will not be visible unless you scroll down to it.
- Fortunately, jQuery Mobile has a method of permanently positioning the menu to the bottom of the screen, regardless of the content on the page.
- This is a *fixed toolbar*. It means that the user can scroll up and down and the menu will still be visible.
- Example

Navigation Bars

- A slightly different approach to the one we have just seen is to use a jQM navigation bar.
- jQM navigation bars are always full screen width and also support highlighting of the active page.
- To create a jQM navigation bar, we use *data-role="navbar"* inside a containing an unordered list of <a> elements.

```
<div data-role="navbar">
  <ul>
   <li><a href="contact.html">Contact</a></li>
   <li><a href="about.html">About</a></li>
   <li><a href="login.html">Login</a></li>
   <li><a href="register.html">Register</a></li>
  <ul>
</div>
```

- Example.

Full-screen Navigation

- In some circumstances (e.g. when we are viewing an image), it would be better that we remove the navigation choices from view entirely.
- In this mode, the navigation disappears after a few seconds, and only reappears when the user taps the screen.
- To create fullscreen mode, we use the *data-fullscreen* property in the header or footer element.

```
<div data-role="footer" data-position="fixed" →
data-fullscreen="true">
```

- Example

Persistent Navigation

- One of the neat features that jQM provides is the ability to persist toolbars across pages.
- The effect of this is that the toolbar will be visible during transitions.
- To create a *persistent toolbar*:
  - The footer with accompanying navbar must be on each page.
  - The footer with navbar must have the same *data-id* (not id) value on each page.
  - The active page in the navigation must use two CSS classes: *ui-state-persist*, and *ui-btn-active.*
  - The persistent footer feature must be used.

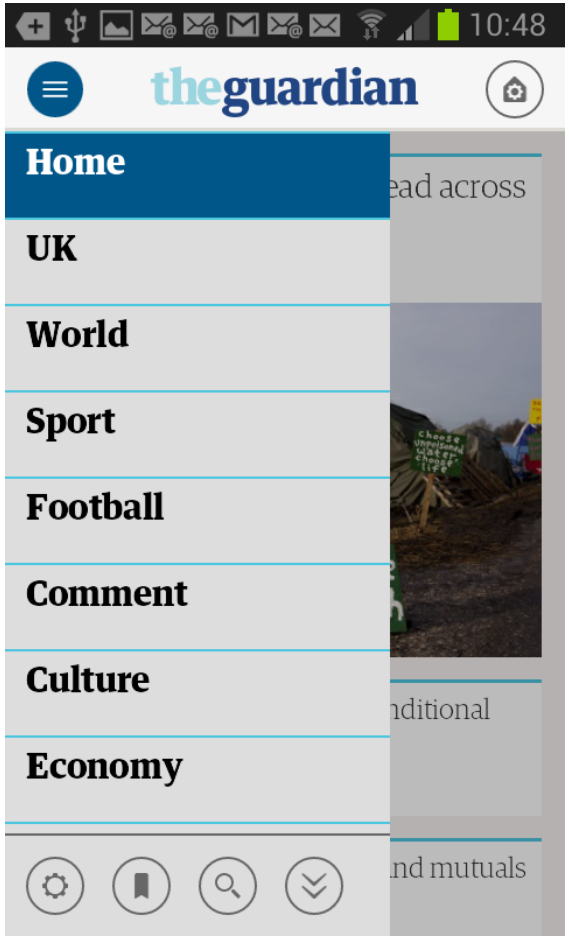
Persistent Navigation

```
<div data-role="footer" data-position="fixed" data-id="footernav">
 <div data-role="navbar">
   <ul>
     <li><a href="contact.html">Contact</a></li>
     <li><a href="about.html">About</a></li>
     <li><a href="login.html" class="ui-state-persist →
     ui-btn-active">Login</a></li>
     <li><a href="register.html">Register</a></li>
  </ul>
  </div>
</div>
```

- Example

Panel Navigation

- Another alternative often used in mobile app navigation is *panel* navigation.
- Panel navigation is used to organise navigation where there are :
  - A lot of elements.
  - Several different navigation schemas in play at the same time (e.g. Facebook).
- A panel displays until the user touches a close option, or touches outside the panel.



Creating a jQM Panel

- To create a panel in jQM, we add a element with a *data-role* of *panel* before the header of a jQM page.
- To be able to refer to the panel from elsewhere in the page, we also need to give the panel an *id* property and value.

```
<div data-role="page" id="first" data-theme="a">
   <div data-role="panel" id="navpanel">
        <!-- panel content goes here -->
   </div>
 . . . . .
</div>
```

Opening a Panel

- We should use a three bar menu icon to indicate a panel is available. 
- To create a three bar menu icon, we first create a standard link. Then we add a *ui-btn* class with the additional presentation details for changing the link to a menu icon (e.g. *ui—icon-bars*).

```
<a href="#navpanel" class="ui-btn ui-corner-all →
ui-icon-bars ui-btn-icon-notext"></a>
```

- Example

Panel Display Options

- By default, a panel appears from the left, and pushes the page content to the right. However, we can change this behaviour, by specifying one or both of *data-display* and *data-position.*
- *data-display* specifies the way the panel appears. Apart from the default, values include:
  - *overlay* (the panel appears on top of the page contents)
  - *push* (animates both the panel and page at the same time)
- *data-position* specifies the direction from which the panel will appear (e.g. left (default), right).
- Example

Panel Positioning

- A panel will be displayed with the *position:absolute* CSS property, meaning it will scroll with the page.
- You can set a panel to *position:fixed*, so that its contents will appear no matter how far down the page you have scrolled, by adding the *data-position-fixed="true"* attribute to the panel.
- Example (Compare this to previous example).

Styling Panels

- Panel styles can be themed or otherwise customized (like the rest of jQM).
- However, be careful of customizing the panel width to a fixed or generalized width, as this will override jQM's ability to calculate the optimum width based on device screen size.
- There is no problem with customizing other aspects of the panel, e.g. font styles, background, border, etc.
- Example.

Adding Panel Navigation

- We can add navigation elements to a panel using a standard listview.

  Example

- Note that jQuery Mobile does not support nested listviews.
- To include a nested menu you will need to use a jQM plugin, or style the menu yourself.