Birkbeck
UNIVERSITY OF LONDON

The jQuery Mobile Framework

Mobile Application Development

Session 3

jQuery Mobile

- jQuery Mobile is a touch-optimized GUI framework for developing mobile web applications.
- It is an extension of jQuery core library, making it relatively straightforward to learn for anybody who already knows jQuery.
- It comprises a single JavaScript file and a single mobile optimized CSS file.

jQuery Main Features

- Uses non-obtrusive, semantic HTML 5.
- Cross platform, cross device, cross browser compatibility.
- Fully customizable, using themes.
- UI optimized for touch screens.
- Lightweight (12kb in compressed form).
- Built on progressive enhancement principles.
- Full accessibility support.

jQuery Mobile Compatibility

- jQuery Mobile uses a 3-level graded browser support system:
  - A grade (full support, including page transitions, graphics, Ajax page loading, etc.)
  - B grade (full graphical support, apart from Ajax)
  - C grade (basic HTML experience, User will see only plain HTML).
- The full list of A, B and C level browsers can be found at the jQuery Mobile site.

Progressive Enhancement

- jQuery Mobile is based on a *progressive enhancement* approach to site design.
- Progressive enhancement starts with a basic experience which is accessible to all browsers.
- It then adds incremental layers of complexity for more sophisticated browsers.
- This means that whatever browser a visitor is using, they will have access to the core elements of the application with no non-functioning or (unexplained) missing elements.

Accessibility

- Because it is built on standard-compliant, semantic, HTML 5 and unobtrusive JavaScript, jQuery mobile is fully compatible with the W3C's WAI/ARIA specifications.
- For A-Grade browsers, many of the components in jQuery Mobile use techniques such as focus management and keyboard navigation.

Getting and Using jQuery Mobile

- jQuery Mobile can be downloaded from the jQuery Mobile Site or linked to via a CDN (e.g. Google).
- The linked to or downloaded files must be referenced in the order shown in the snippet below with the jQuery mobile file coming after the CSS file and the jQuery core library in the <head> of your HTML file.

```
<head>
   <meta charset="utf-8">
   <title> Mobile App Development</title>
   <link rel="stylesheet" href="//ajax.googleapis.com/ajax/libs/ →
    jquerymobile/1.4.3/jquery.mobile.min.css" />
   <script src="//ajax.googleapis.com/ajax/libs/jquery/2.1.1/ →
    jquery.min.js"></script>
   <script src="//ajax.googleapis.com/ajax/libs/jquerymobile/1.4.3/ →
    jquery.mobile.min.js"></script>
</head>
```

Getting and Using jQuery Mobile

Birkbeck UNIVERSITY OF LONDON

- There pros and cons using a CDN for jQuery and jQuery Mobile.

Pros	Cons
Dedicated hosts, like Google, generally have faster servers.	New version may not be compatible with your application causing it to break.
Browsers can download a certain number of assets per web site concurrently. As CDN is on another site it is not included in that limit.	No control over host server problems.
Automatically upgrade to the latest version of library.	Will not work in offline mode.

- As a rule of thumb, if offline access is not required, you should use a CDN.

Testing HTML 5 Apps

- Because there are so many device types, platforms and screen size combinations, testing mobile apps is far more complex than testing conventional desktop apps.
- To facilitate mobile app testing, there are a number of approaches we can use.
  - Simulators
  - Emulators
  - Remote labs
  - In-situ on actual devices

Testing HTML 5 Apps

- *Emulators* reproduce the hardware of a particular device and generally give very good rendering and performance emulation. Most manufacturers provide a platform specific emulator for their devices.
- *Simulators* reproduce some of the behaviour of a device. But, they do not work over a real version of a device operating system, and rendering is generally less precise.
- Neither emulators or simulators are able to adequately simulate touch in desktop environments.
- *Remote labs* are web services that allow us to use real devices remotely. They provide access to just about every device and platform, but those that provide access to different platforms are as a rule subscription based.

Testing HTML 5 Apps

- By far the best way to test a web application is directly on physical devices and platforms.
- Testing on actual devices allows us to see exactly how touch based interactions and page transitions will work.
- The problem with this approach, however, is that:
  - It demands access to multiple devices
  - It requires hardware with the ability to connect to those devices
  - It is expensive
- In reality, unless you work for a company that is willing to make substantial investment in hardware, you are unlikely to be able to test your apps in more than a few different devices.

Resources for Mobile App Testing

- Google Chrome Canary
- Opera Mobile Classic Emulator
- Adobe Edge Inspect
- Mobile Phone Emulator
- Device Anywhere

- For testing on this module you will use a combination of:
  - Google Chrome Canary
  - Mobile Phone Emulator
  - Simply uploading our apps to the DCSIS Titan web server, and viewing them in your phones and/or tablets.

jQuery Mobile Basics

- To create a basic jQuery mobile web app we start with a standard HTML 5 document template.
- For this we need to add our CDN links to the jQuery and jQuery mobile libraries, as well as to our jQuery Mobile CSS file.
- We also need to specify the *viewport* for the page to override the browser specific viewport settings.
- The viewport should be:
  - set at an initial scale of 1. This means the content is not zoomed in on initial loading
  - set with the content width set to the device screen width
- Viewport reference

jQuery Mobile Basics

```
<!doctype html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title> Mobile App Development</title>
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="stylesheet" href="//ajax.googleapis.com/ajax/libs/ →
    jquerymobile/1.4.3/jquery.mobile.min.css" />
   <script src="//ajax.googleapis.com/ajax/libs/jquery/2.1.1/ →
    jquery.min.js"></script>
   <script src="//ajax.googleapis.com/ajax/libs/jquerymobile/1.4.3/ →
    jquery.mobile.min.js"></script>
</head>
```

jQuery Mobile Pages

- jQuery Mobile's central abstraction is the use of multiple pages inside a single HTML document.
- jQM pages are elements inside a HTML document that have been enhanced into *page widgets*.
- Only one page is visible in the browser window at any time.
- Upon navigation, the currently visible page is hidden, and another page is shown. Moving from one page to another is accomplished using a *transition*.
- Example

jQuery Mobile Page Terminology

- Internal pages
  - elements enhanced into a page *widget*.
  - Each internal page has an *id* attribute to distinguish it from other internal pages.
- External pages
  - jQuery Mobile HTML documents located at the same domain.
  - External pages can only contain *one* internal page.
- Absolute external pages
  - Non-jQuery Mobile HTML documents located at other domains.

Internal Pages

- Internal jQM pages are created using a combination of tags and *custom data attributes*.
- Custom data attributes are a HTML 5 specific feature which allow developers to define their own attributes while still maintaining valid HTML.
- The key custom data attribute of jQM, is *data-role*
- *data-role* allows us to specify a specific role for an element on a page. This includes:
  - page
  - header
  - content
  - footer
  - button
  - listview
- Example

Internal Pages

- New internal pages in jQM are created by adding new tags, and assigning them a *data-role* of *page*.

```
<body>
     <div data-role="page" id=”main”>
    </div>
      <div data-role="page" id=”second”>
    </div>
</body>
```

- When the HTML document first loads, only the first page (unless specified) will be visible in the browser.

Internal or External Page?

- The advantage of using internal pages is that only a single HTML document needs to be cached, meaning download time for an application is faster than if several HTML documents were used.
- Theoretically, we can add as many pages to a single HTML document as we like. However, if we keep adding pages to a single document, the DOM size will become unwieldy, and initial download of the document will be slowed
- Ideally, we should aim to include pages within the same primary document that are very likely to be used frequently. Other pages should be created as external jQM pages.

Pre-fetching Pages

- We can speed up the loading of external pages by prefetching them into the DOM.
- A prefetched page will be instantly available to a user.
- Pages are prefetched by adding *data-pre-fetch* to all the links you want to pre-fetch.

```
<a href="prefetchedpage.html" data-prefetch> ... </a>
```

- An advantage of prefetching a page is that the user doesn't see the Ajax loading message when visiting the pre-fetched page.
- Prefetching pages creates additional bandwidth load and, so it is better to use this feature only in situations where it is likely that the pre-fetched page will be visited.
- Example

Absolute External Links

- When linking to an external site or document using jQM, we need to bear in mind that the site may not be optimized for mobile consumption, and thus may be out-of-synch in terms of design and functionality with the rest of your application. Or may not be a mobile optimised site at all.
- One way around this is to force the incoming document to open in a new browser window.

```
<a href="http://www.accexternalsite.com" →
target="_blank">Access</a>
```

- Example

jQM Navigation Basics

- Links to internal pages are created using an anchor as the *href* attribute value.

```
<a href="#services">Services</a>
```

- Links to external pages are created using a normal *href* attribute value.

```
<a href="access.html">Access</a>
```

- Example

jQuery Mobile Navigation

- jQuery Mobile replaces the browser's standard HTTP navigation with Ajax-based navigation. It overrides the browser's default link handling behaviour.
- When a link is clicked, the event is intercepted by the framework and an AJAX request rather than an HTTP request is issued.
- When the AJAX request is complete, jQuery Mobile injects the code for the requested page into the DOM of the original page, thus avoiding the need for a full page reload.
- When the code for the requested page is in the DOM, a *page transition* is applied, and the new page is animated into view. Example
- Note that transition behaviours cannot apply to absolute external links.

jQuery Mobile Navigation

- To create meaningful URLs and manage user history, jQM uses the .*history* object, or in older browsers hash tag navigation.
- Thus, if we have an application with two internal pages. The URL for the first and second pages will be as follows.

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

- Note that he use of hash tags in jQM links and URLs precludes the use of pages with traditional style anchor links.

jQuery Mobile Page Structure

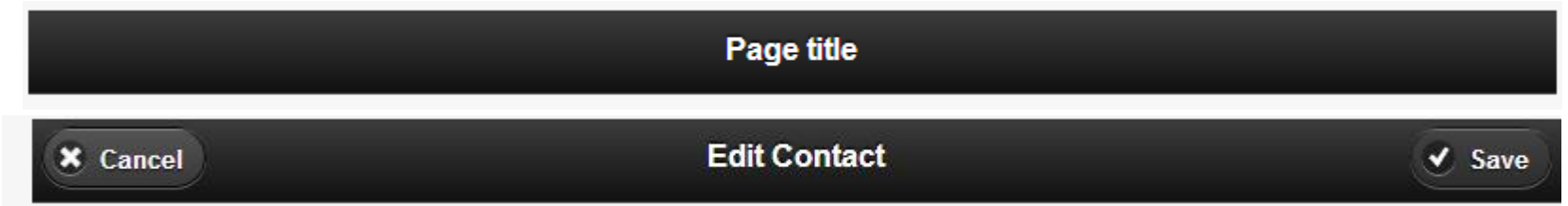
- The displayed sections of a jQuery Mobile page are given three separate roles: header, content and footer
- Roles are assigned to these sections using the *data-role* attribute.

```
<div data-role="page" id="foo">
      <div data-role="header">
      </div>
      <div data-role="content">
      </div>
      <div data-role="footer">
      </div>
</div>
```

- The actual roles of these sections are semantically self-explanatory.
- However, there are some notable differences between how they are used in desktop applications, and how the are used in jQM apps.

Page Header

- The header contains the page title text and optional buttons (e.g. back, or home) positioned to the left and/or right of the title for navigation or functionality.
- Primary navigation is generally NOT positioned in the header.

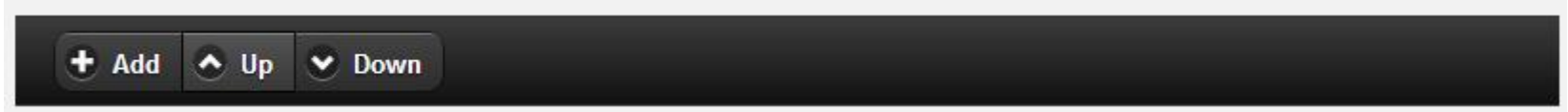


- The page title text is normally an <h1> heading element.
- However, for better semantics, a <h1> element can be used on the landing page and <h2> elements on secondary, internal, pages.
- Note that where header content is > 15 characters long it will be cut, and extra characters replaced by an ellipsis (e.g. *Introducing P..*).

Page Footer

- Footer content is generally wrapped inside an <h4> element.

- Unlike in desktop apps, the page footer is often used for global navigation elements rather than simply for information.
- The main difference is that the footer is meant to be less structured than the header allowing greater flexibility in positioning elements.
- Unlike for headers, however, the framework does not automatically create dedicated slots for buttons to the left or right. They are simply arranged inline.



Content Section

- The content section of a jQM app is fundamentally different from the content section of a desktop app.
- The content section is used for primary and secondary navigation choices, as well as simple content.
- Where the content section is used for actual content, then the way content is structured has to be much more carefully thought out due to screen size restrictions.
- The primary method of structuring extended content in jQM apps is the use of collapsible sets (accordions). Example.