Customizing jQuery Mobile Apps

Mobile Application Development

Session 8

jQuery Mobile Themes

- jQuery Mobile uses a *theming* system to allow customization of mobile apps.
- In jQuery Mobile, a theme is *a unified visual design* applied across the interface. In technical terms, it is simply a collection of CSS code applied to jQM widgets.
- A jQM theme specifies everything from fonts to drop shadows to colours.
- Themes allow for up to 26 unique *colour swatches* per theme, creating the ability for almost unlimited variety in designs.

jQuery Mobile Swatches

- In keeping with the idea of separating layout from color and texture, a jQuery Mobile theme can have multiple *swatches*.
- A swatch is a *unified color concept* governing the colours of background, text, shadows, iconography, etc.
- The default jQuery Mobile theme (> v 1.4) includes two swatches (a and b). Each swatch provides different visual emphasis, with swatch *a* being the most visually emphatic and accessible.

jQuery Mobile Themes

- The default jQuery Mobile themes are defined in the framework CSS file.
- Custom themes can be created using the ThemeRoller theme generator.
- Changes to existing themes can be made by using custom CSS 3 to override settings in a default or custom theme.

Applying jQuery Mobile Swatches

- Swatches can be set at page level.
- If no swatch is explicitly chosen, the framework will default to swatch *a* (light grey header and footer, white content section, and black font). Example.
- We can a new swatch for a jQM page by setting the *data-theme* attribute in the page to our selected swatch.

```
<div data-role="page" id="home" data-theme="b">
```

- Example

Applying jQuery Mobile Swatches

- Swatches can also be mixed and matched for individual page elements (e.g. header, footer, listview, etc.) to give greater customization to the look and feel of a page.

```
<div data-role="page" id="home" data-theme="b">
  <div data-role="header" data-theme="a">
    <h1>Zedland Hotels</h1>
  </div>
  <div data-role="content">
    <h2>Hotels</h2>
    <ul data-role="listview" data-inset="true" data-theme="a">
```

- Example

Overriding Themes using Custom CSS

- jQM allows us to override the CSS from default or custom themes.
- This is done using custom CSS files which set the value of the jQM classes associated with jQM widgets (e.g. ui-btn, ui-listview, etc.).
- The following snippets sets the margins for specific listviews and the corner radius for all widgets.

```
.ui-content .ui-listview{
margin: 2.5% 5% 5% 5%;
}

.ui-corner-all {
border-radius: 10px !important;
}
```

- For the override to work, the custom CSS must be included after the jQM CSS file. Example.

Setting the Theme Globally

- Setting the theme for a single page will not affect the appearance of any other internal or external pages in an application.
- To apply a theme across an application we need access the jQuery Mobile framework default configurations and change these.
- This needs to be done programmatically using jQM *mobileinit* event.

Setting the Theme Globally

- *mobileinit* is triggered as part of the library's loading process. It allows us the opportunity to access and tweak jQuery Mobile's configurations at framework level.

```
$(document).bind('mobileinit', function () {
});
```

- Once we have access to the global configurations we can make changes to page widgets, such as page, listview, collapsible set, etc. We do this by accessing the *options* of the *prototype* of the widget. Example.

```
$(document).bind('mobileinit', function () {
   $.mobile.page.prototype.options.theme  = "d";
});
```

Making Global Changes

- When we use the *mobileinit* event to make global changes to our app, we need to make sure our configuration script runs before the jQuery Mobile library script.
- In the example below, our configuration script is placed before the jQM library in the <head> of the document. Example.

```
<script src="config.js"></script>
<script src="//ajax.googleapis.com/ajax/libs/ →
jquerymobile/1.4.3/jquery.mobile.min.js"></script>
```

Making Global Changes

- As well as allowing us to make changes to the default theme, *mobileinit* also allows us to make global changes to any other jQM widget, as well as to other elements of the framework such as transitions. Example.

```
$(document).bind('mobileinit', function () {
     $.mobile.defaultPageTransition = 'slidefade';
     $.mobile.page.prototype.options.theme  = "d";
});
```

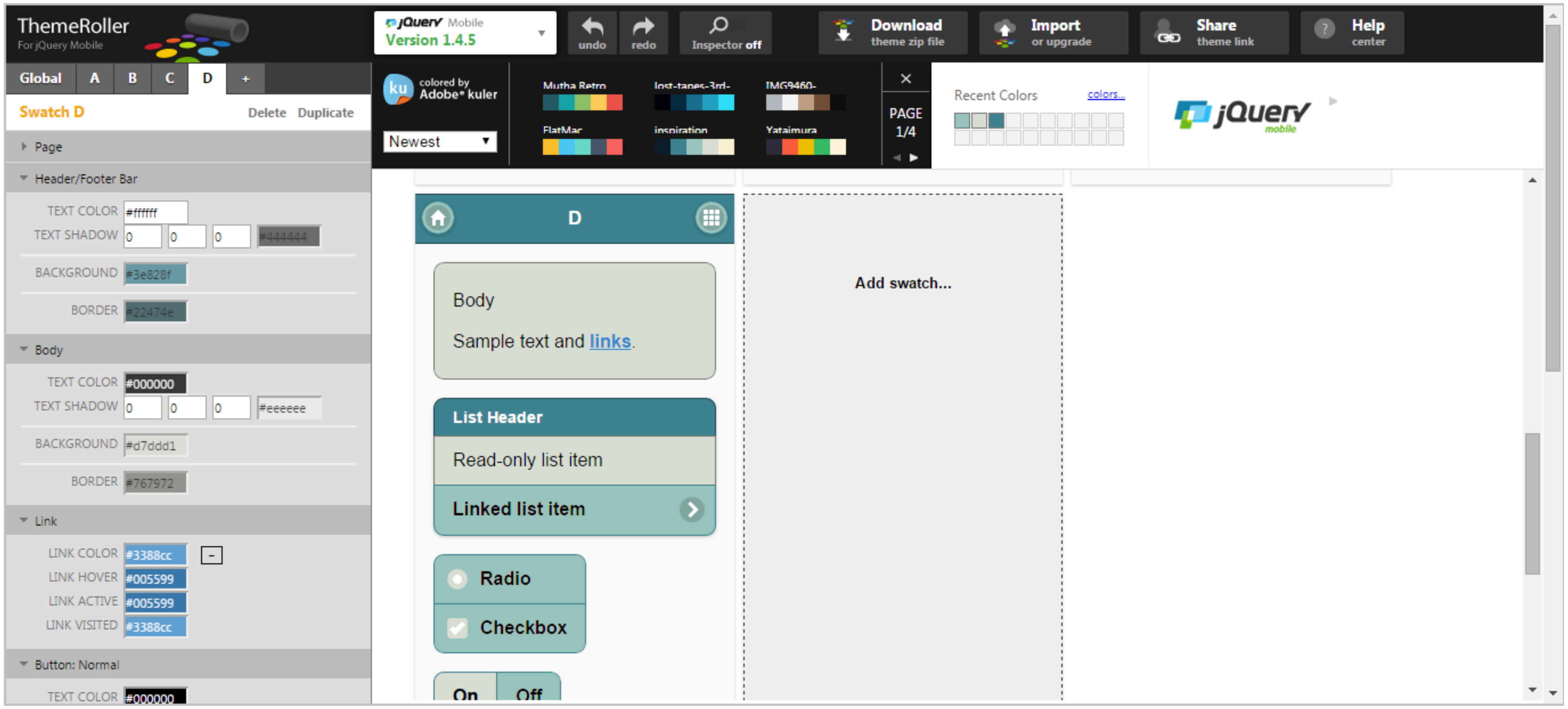
Custom Themes

- The default theme and the two swatches that come with it limit the scope of what we can do with the look and feel of jQM applications.
- However, this is easily overcome as jQM allows us to create and use custom themes.
- The main theme creator tool for jQM is ThemeRoller.

Custom Themes with ThemeRoller

- ThemeRoller for jQuery Mobile is a web app that allows a user to configure settings for colours, background textures, and fonts — including variations for default, hover and active states — for the component pieces of all jQuery Mobile UI widgets.
- Themes generated using ThemeRoller can be downloaded as a ZIP file and included in an application where they can be used in place of the jQM default themes.

Custom Themes with ThemeRoller



Swatch D Delete Duplicate

Page

Header/Footer Bar

TEXT COLOR #ffffff
TEXT SHADOW 0 0 0 #444444
BACKGROUND #3e828f
BORDER #22474e

Body

TEXT COLOR #000000
TEXT SHADOW 0 0 0 #eeeeee
BACKGROUND #d7ddd1
BORDER #767972

Link

LINK COLOR #3388cc
LINK HOVER #005599
LINK ACTIVE #005599
LINK VISITED #3388cc

Button: Normal

TEXT COLOR #000000

Custom Themes with ThemeRoller

- To apply a ThemeRoller theme, we link to the downloaded CSS files for the newly created styles and icons in the <head> of each page in our application.
- Note that the <link> tags for the CSS files must be positioned *before* the jQM main CSS file.

```
<link rel="stylesheet" href="zedland-custom.min.css" />
<link rel="stylesheet" href="jquery.mobile.icons.min.css" />
<link rel="stylesheet" href="http://code.jquery.com/mobile/→
1.4.5/jquery.mobile.structure-1.4.5.min.css" />
```

Applying Custom Themes

- Once we have included our custom theme in our application files, we then need to apply that theme.
- To apply the theme globally, we need to apply it at framework configuration time using *mobileinit*.
- In the following example, the theme was generated under letter *d* in ThemeRoller, and is applied globally using that letter. Example.

```
$(document).bind('mobileinit', function () {
  $.mobile.page.prototype.options.theme  = "d";
});
```