CS338 Developing User Interfaces

Lab 4: Introduction to jQuery Mobile

So, although we did all this work developing the Bike Drexel app using HTML5, CSS, and JavaScript, in this lab we introduce jQuery Mobile, a framework that will do most of the work for us.

In this, labs we will develop our web app Bike Drexel from scratch using JQuery Mobile.

In this lab you will learn to:

- Create a multipage webapp
- Add buttons to the header and content elements
- Display dialogs and popups
- Link external and internal pages
- Design your own themes
- Selecting and implementing a navigation pattern

Prerequisites

Before you start, login to tux and create a new directory called “Lab4” in your public_html/cs338 directory. Grab all the files from Lab4 on BBLearn and save them in the Lab4 project directory you just created.

What is jQuery Mobile?

jQuery Mobile (http://jquerymobile.com/) is a HTML5-based mobile framework written in JavaScript that is designed to make responsive websites and apps that are accessible on all smartphone, tablet and desktop devices.

jQuery Mobile put together 3 things:

1. Javascript API and library
2. CSS and themes (natively styled)
3. Effects

Pages:

jQuery Mobile uses a very simple and powerful approach to define the content of the webapp. The main unit of the framework is a `page`. A `page` is just a `div` element with a specific role. One HTML document can host one page or many pages inside the same file. This is a new concept that is different from simple HTML5.

Roles:

jQuery Mobile uses standard HTML markup, such as the `div` tag. To define what the framework should do with that `div`, we define a `role`. A `role` in the framework is defined using the attribute `data-role`. For example, `<div data-role= “page”>.`

The main roles available in jQuery are defined in the following table:

Role	Description	URL
page	Defines a page, the unit the jQuery Mobile uses to show content	http://demos.jquerymobile.com/1.4.5/pages/
toolbar	Toolbars (header/footer) of a page	http://demos.jquerymobile.com/1.4.5/toolbar/
grid	Grid (columns) for content of a page	http://demos.jquerymobile.com/1.4.5/grids/
navbar	Defines a navigation bar, typically inside a header	http://demos.jquerymobile.com/1.4.5/navbar/
button	A visual button	http://demos.jquerymobile.com/1.4.5/button-markup/ http://demos.jquerymobile.com/1.4.5/button/
controlgroup	Renders a group of buttons	http://demos.jquerymobile.com/1.4.5/controlgroup/
collapsible	Collapsible panel of content inside a page	http://demos.jquerymobile.com/1.4.5/collapsible/
listview	Content of multiple items as a list	http://demos.jquerymobile.com/1.4.5/listview/
dialog	Dialog page	http://demos.jquerymobile.com/1.4.5/pages-dialog/

Mobile page structure

jQuery Mobile site must start with an HTML5 'doctype' to take full advantage of all of the framework's features. In the head, references to jQuery, jQuery Mobile, and the mobile theme CSS are all required to start things off:

```
<!DOCTYPE html>
<html>
	<head>
	<title>Page Title</title>
	<link rel="stylesheet"
href="http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.css" />
	<script src="https://code.jquery.com/jquery-1.11.1.min.js"></script>
	<script src="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
1.4.5.min.js"></script>
</head>
```

```
<body>

...

</body>
</html>
```

Inside the `<body>` tag, each view or page on the mobile device is identified with an element (usually a div) with the `data-role="page"` attribute:

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

Within the page container, any valid HTML markup can be used, but for typical pages in jQuery Mobile, the immediate children of a page are `divs` with data-roles of: `header, content`, and `footer`.
Here is a skeleton of a page:

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

Complete single page template

Putting it all together, following is the standard boilerplate `page` template you should start with.

Step 1: Create a new HTML 5 file called `SinglePage.html`. Copy the following jQuery Mobile single page template to your file. Save your file and copy it to your tux account. Test your file.

If all worked well you should be able view your first jQuery Mobile web app.

```
<!DOCTYPE html>
<html>
       <head>
       <title>Page Title</title>
       <link rel="stylesheet"
href="https://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.css" />
       <script src="https://code.jquery.com/jquery-1.11.1.min.js"></script>
       <script src="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
1.4.5.min.js"></script>
</head>
<body>
```

```
<div data-role="page">

	<div data-role="header">
	<h1>Page Title</h1>
	</div><!-- /header -->

	<div data-role="content">
	<p>Page content goes here.</p>
	</div><!-- /content -->

	<div data-role="footer">
	<h4>Page Footer</h4>
	</div><!-- /footer -->
</div><!-- /page -->

</body>
</html>
```

External page linking

jQuery Mobile automates the process of building Ajax powered sites and applications. Remember, AJAX = Asynchronous JavaScript and XML. It is a technique for creating fast and dynamic web pages. AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page. Classic web pages, (which do not use AJAX) must reload the entire page if the content should change. To learn more about AJAX go here: http://www.w3schools.com/ajax/ajax_intro.asp

By default, when you click on a link that points to an external page (ex. http://drexel.edu/cci/departments/computer-science/), the framework will parse the link's href to formulate an Ajax request and displays the loading spinner.

If the Ajax request is successful, the new page content is added to the DOM, all mobile widgets are auto-initialized, then the new page is animated into view with a page transition.

Step 2: Add a hyperlink to test.html to the `content` div and test your page. (Make sure SinglePage.html has been saved to your Lab4 directory).
e.g. `<a href="test.html">Here is a link<a/>`
Now, change the href attribute to a non-existing file (e.g. testtt.html). Save and test your page. How does the application respond to the failed request?

Page Transitions

The jQuery Mobile framework includes a set of CSS-based transition effects that can be applied to any page link with Ajax navigation. Here is a list of jQuery Mobile transitions:

- fade
- pop
- flip
- turn
- flow
- slidefade
- slide
- slideup
- slidedown
- none

Step 3: By default, the framework applies a fade transition. To set a custom transition effect, add the data-transition attribute to the link:

```
<a href="test.html" data-transition="pop">I'll pop</a>
```

Experiment with the different transitions listed above. Consider when it is appropriate to use each of the transitions.

To learn more about transitions go here: http://demos.jquerymobile.com/1.4.5/transitions/

Multipage template

A single HTML document can contain multiple pages that are loaded together by stacking multiple `divs` with a `data-role` of `page`. Each page block needs a unique ID that will be used to link internally between pages (e.g. `<a href="#foo"> link </a>`). When a link is clicked, the framework will look for an internal page with the ID and transition it into view.

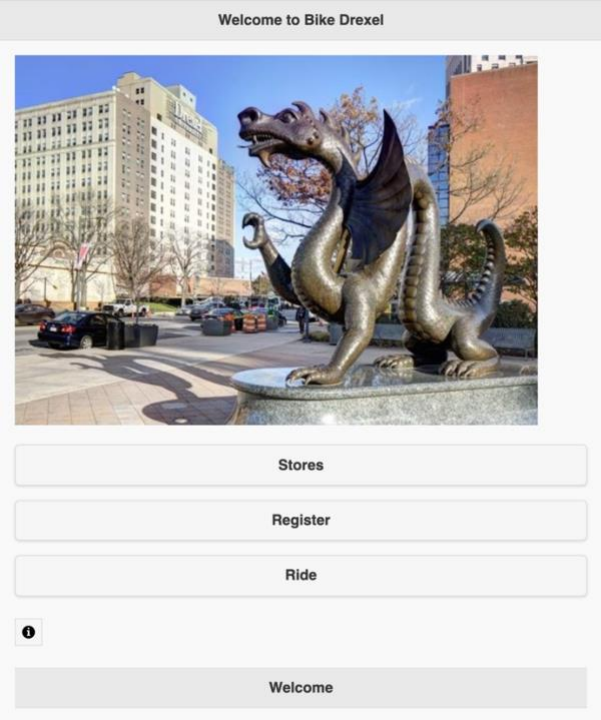
Here is an example of a 2 page site built with two jQuery Mobile `divs` navigated by linking to an ID placed on each page wrapper.

```
<body>

<!-- Start of first page -->
<div data-role="page" id="foo">

	<div data-role="header">
	<h1>Foo</h1>
	</div><!-- /header -->
```

```
	<div data-role="content">
	<p>I'm first in the source order so I'm shown as the page.</p>
	<p>View internal page called <a href="#bar">bar</a></p>
	</div><!-- /content -->

	<div data-role="footer">
	<h4>Page Footer</h4>
	</div><!-- /header -->
</div><!-- /page -->


<!-- Start of second page -->
<div data-role="page" id="bar">

	<div data-role="header">
	<h1>Bar</h1>
	</div><!-- /header -->

	<div data-role="content">
	<p>I'm second in the source order.</p>
	<p><a href="#foo">Back to foo</a></p>
	</div><!-- /content -->

	<div data-role="footer">
	<h4>Page Footer</h4>
	</div><!-- /footer -->
</div><!-- /page -->
</body>
```

Step 4: Next we will use jQuery Mobile to re-implement our **Bike Drexel** webapp. **Based on the above example**, create and edit index.html so that it contains 4 pages: Welcome, Stores, Register, and Ride. Edit the titles, headers, and footers so that they reflect the content of our webapp. Make sure to put the jQuery header step 1.

Step 5: Add links from the welcome page to each of the other pages. You can also add an image or a banner (files provided in the project folder). Test your app. It should look something like this:



Adding Back buttons
If you use the attribute `data-rel="back"` on an anchor tag (`<a>`), any clicks on that anchor will mimic the back button, going back one history entry and ignoring the anchor's default `href`. Clicking the back button will also invoke a reverse transition.

Step 6: Add a back button to the pages: stores, register, and ride by adding the following code to their header:

```
<a data-icon="back" data-rel= "back" back-btn="true">Back</a>
```

Header Structure
The header is a toolbar at the top of the page that usually contains the page title text and optional buttons positioned to the left and/or right of the title for navigation or actions.

The title text is normally an H1 heading element but it's possible to use any heading level (H1-H6) to allow for semantic flexibility.

In the header there are slots for buttons on either side of the text heading. Each button is typically an anchor (<a>) element, but can be a button element as well. To save space, buttons in toolbars are set to inline styling (http://demos.jquerymobile.com/1.2.0/docs/buttons/buttons-inline.html) so the button is only as wide as the text and icons it contains.

The header plugin looks for immediate children of the header container, and automatically sets the first link in the left button slot and the second link in the right. In this example, the 'Cancel' button will appear in the left slot and 'Save' will appear in the right slot based on their sequence in the source order.

```
<div data-role="header">
       <a href="index.html" data-icon="delete">Cancel</a>
       <h1>Edit Contact</h1>
       <a href="index.html" data-icon="check">Save</a>
</div>
```

Buttons automatically adopt the swatch color of the bar they sit in. Later we will introduce a way to change their color and make them visually stand out.

The jQuery Mobile framework includes a selected set of icons most often needed for mobile apps. An icon can be added to a button by adding a `data-icon` attribute on the anchor (`<a>`) specifying the icon to display. For example, try adding the following markup to the header.

```
<a href="index.html" data-role="button" data-icon="delete">Delete</a>
```

Check the following link for a list of data-icons: http://demos.jquerymobile.com/1.4.5/button-markup/

By default, all icons in buttons are placed to the left of the button text. This default may be overridden using the `data-iconpos` attribute to set the icon to the right, top, or bottom of the text. For example, try the markup:

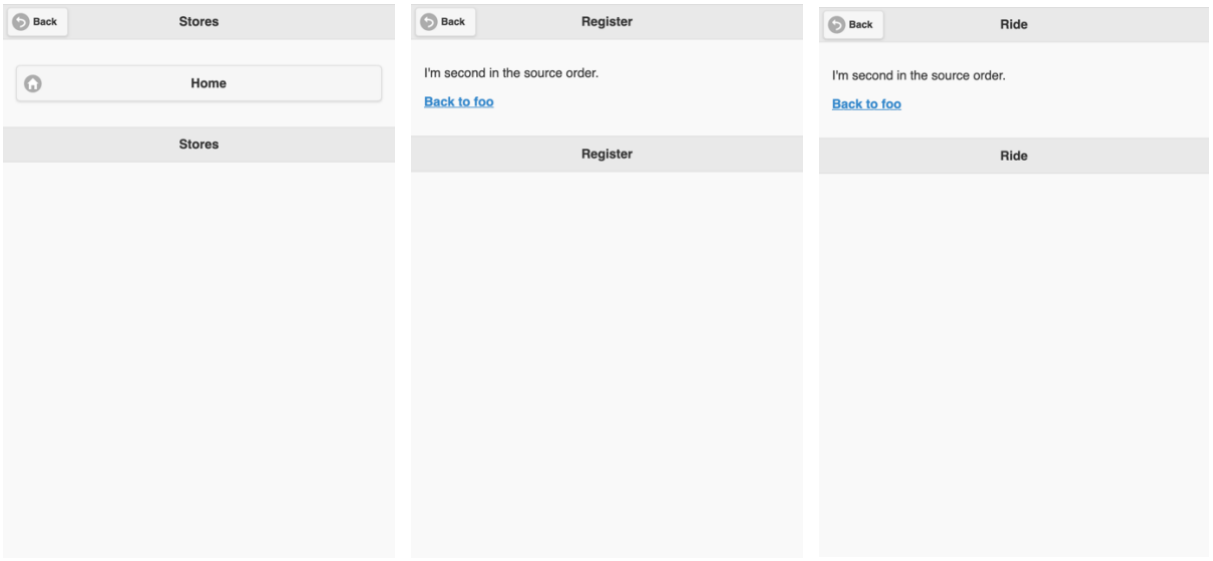
```
<a href="index.html" data-role="button" data-icon="delete" data-
iconpos="right">Delete</a>
```

Check the link above for more information about styling and positioning button icons.

Step 7: Add a button to the header of each of the webapp pages (in addition to the back button added in Step 6). Experiment with different `data-icons` as well as with different `data-iconpos` values.

Step 8: Add a button to the body of the stores page. The button should link to the welcome page and display an appropriate `data-icon`.

Checkpoint:
For now, here is how the pages Stores, Register, and Ride should look like:



Popups

A **popup** consists of two elements: the screen, which is a transparent or translucent element that covers the entire document, and the container, which is the popup itself. To create a popup, a popup div **has to be nested inside the same page as the link** and the `data-role="popup"` attribute need to be added to the `div` with the popup contents.

Then a link with the `href` set to the id of the `popup div`, and the attribute `data-rel="popup"` tells the framework to open the popup when the link is clicked.

Step 9: Add the following anchor tag and popup div to index.html and test your application.

```
<a href="#popupBasic" data-role="button" data-rel="popup">Open Popup</a>

<div data-role="popup" id="popupBasic">

      <p>This is a completely basic popup, no options set.<p>

</div>
```

How do you close the popup?

The jQuery framework makes it easy to create a variety of different popups by adding in elements and through simple styling. **Here are a few real-world examples that combine various settings and styles**:

A tooltip popup

```
<p><a href="#popupInfo" data-rel="popup" data-transition="pop" class = "my-tooltip-
btn ui-btn ui-alt-icon ui-nodisc-icon ui-btn-inline ui-icon-info ui-btn-icon-
notext">Learn more</a></p>

<div data-role="popup" id="popupInfo" class="ui-content" data-theme="a" style="max-
width:350px;">Tooltip</div>
```

A form popup

```
<div data-role="popup" id="popupLogin" data-theme="a" class="ui-corner-all">
      <form>
      <div style="padding:10px 20px;">
            <h3>Please sign in</h3>
             <label for="un" class="ui-hidden-accessible">Username:</label>
<input type="text" name="user" id="un" value="" placeholder="username" data-theme="a"
/>

             <label for="pw" class="ui-hidden-accessible">Password:</label>
<input type="password" name="pass" id="pw" value="" placeholder="password" data-
theme="a" />

             <button type="submit" data-theme="b">Sign in</button>
      </div>
      </form>
</div>
```

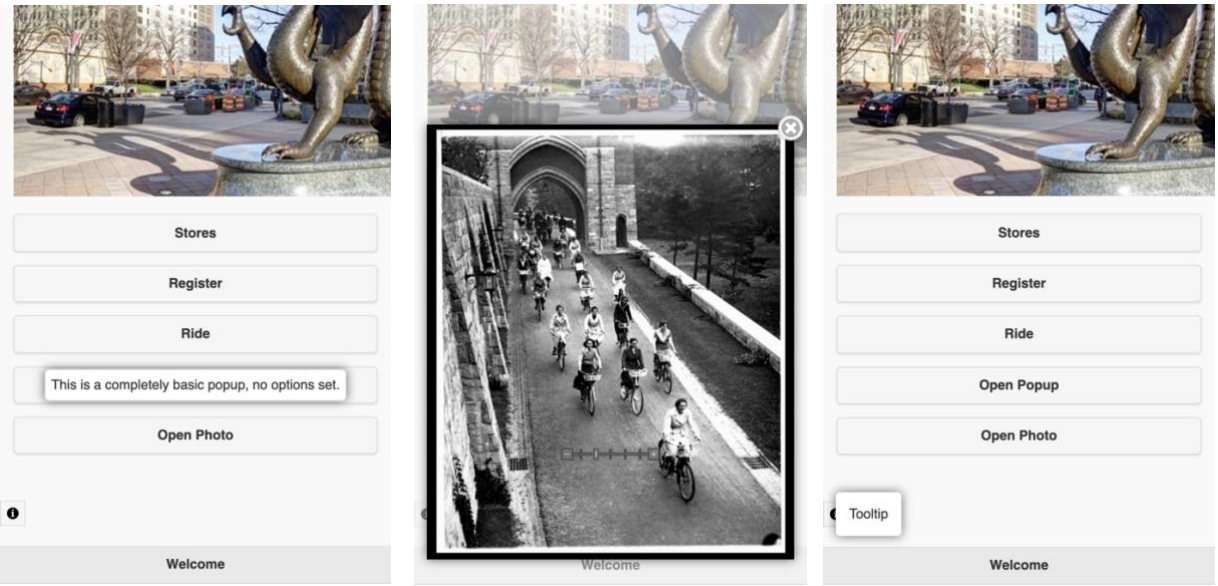
An image popup

```
<div data-role="popup" id="popupPhoto" data-overlay-theme="a" data-theme="d" data-
corners="false">
      <a href="#" data-rel="back" class="ui-btn ui-corner-all ui-shadow ui-btn-a ui-
icon-delete ui-btn-icon-notext
ui-btn-right">Close</a> <img class="popphoto" src="bikes.jpg.png" alt="Students
riding">
</div>
```

Step 10: Add anchor tags to index.html to display the above popups. Test your application.

Check out the following link for more information about popups:
http://demos.jquerymobile.com/1.4.5/popup/

Checkpoint: here is how your popups should look like:



Themes

The default theme includes 5 swatches that are given letters (a, b, c, d, e) for quick reference. To make mapping of color swatches consistent across our widgets, we have followed the convention that swatch "a" is the highest level of visual priority (black in our default theme), "b" is secondary level (blue) and "c" is the baseline level (gray) that we use by default in many situations, "d" for an alternate secondary level and "e" as an accent swatch. Themes may have additional swatches for accent colors or specific situations. For example, you could add a new theme swatch "f" that has a red bar and button for use in error situations. Learn more about Themes in [this page (http://demos.jquerymobile.com/1.1.2/docs/api/themes.html)](http://demos.jquerymobile.com/1.1.2/docs/api/themes.html).

Most theme changes can be done using [ThemeRoller (https://themeroller.jquerymobile.com/)](https://themeroller.jquerymobile.com/), but it's also simple to manually edit the base swatches in the default theme and/or add additional swatches by editing the theme CSS file. Just copy a block of swatch styles, rename the classes with the new swatch letter name, and tweak colors as you see fit.

Navigation

An important aspect of creating a usable web app is selecting and implementing a navigation pattern. Here are some common navigation patterns that are supported by JQueryMobile:

- *Tree Drill Down*: this pattern allows for filtering through information that is grouped typically in the form of a list.
- *Header Menus*: this pattern provides tabs, buttons, or tools in the header to assist in navigation.
- *Footers and Nav Bars*: this pattern allows the users to move horizontally across the app.
- *Overlays and Modals*: this pattern provides an overlay of information instead of requiring the user to transition to a different page. This helps the user to maintain context.

Now, we will focus on the Tree Drill Down and Nav Bars navigation patterns.

Tree Drill-Down

Step 11: In index.html add a page `div` to the file with `id="bicycles"` .

Step 12, listview: In index.html, find the page div "welcome". Let's add a list of links to the content `div`. To do so we will add a `<ul>` element with an attribute `data-role="listview"` to the `<div role="content">.`

(Note that by now you should have additional content in this `div`, specifically a set of buttons. This is fine – you should make a decision where to add the list so that it fits nicely with the rest of your HTML content.)

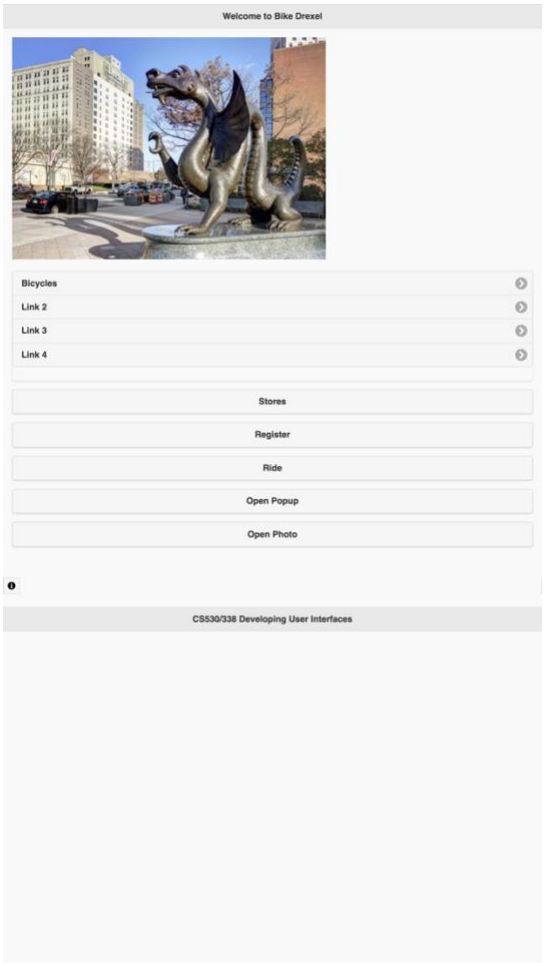
Add the following code to div content:

```
<ul data-role="listview" style="padding-bottom: 25px;">

   <li><a href="#bicycles">Bicycles</a></li>
   <li><a href="">Link 2</a></li>
   <li><a href="">Link 3</a></li>
   <li><a href="">Link 4</a></li>

</ul>
```

For now, only the first link will work.

Step 13, inline list: JQuery Mobile provides various ways to style lists. Since this list is mixed with other HTML content, we will style it as inline list. To do so add the attribute `data-inset="true"` within the `<ul>` element.

Test your application and see how JQuery Mobile renders your code. So far you app should look something like this:



Step 14, full-page list: Next, let's drill down the tree. In index.html find `page div` bicycles.
In this page we will add a full-page list that contains styled text, and thumbnails images.

Add the following code to the `<div role="content">` of the bicycle page:

```
<ul data-role="listview">

   <li>
       <a href="#bike1" class="ui-link-inherit">
            <img src="bicycle5.jpg">
            <h3 class="ui-li-heading">Status</h3>
```

```
            <p class="ui-li-desc">Status description</p>
        </a>
    </li>

     <li>
        <a href="#bike1" class="ui-link-inherit">
             <img src="bicycle5.jpg ">
            <h3 class="ui-li-heading">Status</h3>
            <p class="ui-li-desc">Status description</p>
        </a>
    </li>

    <li>
        <a href="#bike1" class="ui-link-inherit">
             <img src="bicycle5.jpg " >
            <h3 class="ui-li-heading">Status</h3>
            <p class="ui-li-desc">Status description</p>
        </a>
    </li>

       <li>
        <a href="#bike1" class="ui-link-inherit">
             <img src="bicycle5.jpg " >
             <h3 class="ui-li-heading">Status</h3>
             <p class="ui-li-desc">Status description</p>
             </a>
    </li>

</ul>
```

Each row in this list contains both a title (h3 element) and a description (p element) in addition to a thumbnail image. Both the header and the paragraph elements use a special class for styling: ui-li-heading and ui-li-desc.The thumbnail image is defined as an image inside the list item. It doesn't need any special class defined but needs to be **80x80** pixels.

Test your application and see how JQuery Mobile renders your code.

The framework automatically adds a nice arrow at the right side of the row giving the user a hint that the row is touchable. (**Note:** on Chrome Developer tools, you may need to scroll to the right to see them. However, on an actual device, the arrows fit on the screen as expected). This icon can be

changed using `data-icon`: try adding the attribute `data-icon=”info”` to a `<li>` element. If you want to remove the icon use: `data-icon=”false”` in your `<li>` element.

Step 15, aside content: Up to now, our lists only had one column for text content. We can add a second-level column to every row for supplemental information. To do so we can use add a `p` element inside a `li` element. Add the following code at the end of the content of a `li` element:

```
<p class="ui-li-aside">$221</p>
```

Test your application and see how JQuery Mobile renders your code.

Step 16, using count bubbles: a *count bubble* is a circle with a number inside rendered at the right of the row, usually showing how many items are in an interactive row. It can be used to show the number of unread messages, event participants, or any other relevant numeric information in a simple way. To add a count bubble, first remove the *aside content* from your `li` elements, then, insert the following `span` element to the end of the content of any `li` element:

```
<span class=“ui-li-count”>6</span>
```

Test your application and see how JQuery Mobile renders your code.

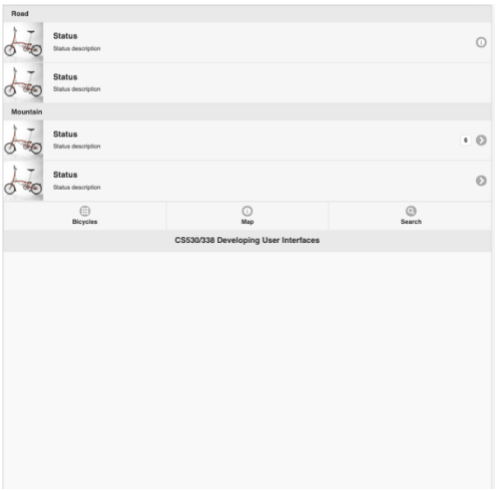
Step 17, visual separators: we can use visual separators to divide a single list into row grouping with their own titles. To do so we will use a new role for `li` elements: `data-role=”list-divider”`.

Add the following `li` element as the first element in your list:

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

Add another divider to the list at an appropriate place. Test your application and see how JQuery Mobile renders your code.

Checkpoint: At this point, the page should look something like this:



Navigation Bar

A navigation bar, also known as `navbar`, is a set of links that can be placed at a toolbar, usually at the footer. A navbar acts as a main navigation method for our webapp. On iOS, the navbar is also called a tab bar.

Note, you should not use navigation bars for action buttons (e.g. Save, Cancel, Search) but as a main structure for your webapp. For action buttons use normal buttons on a header or footer.

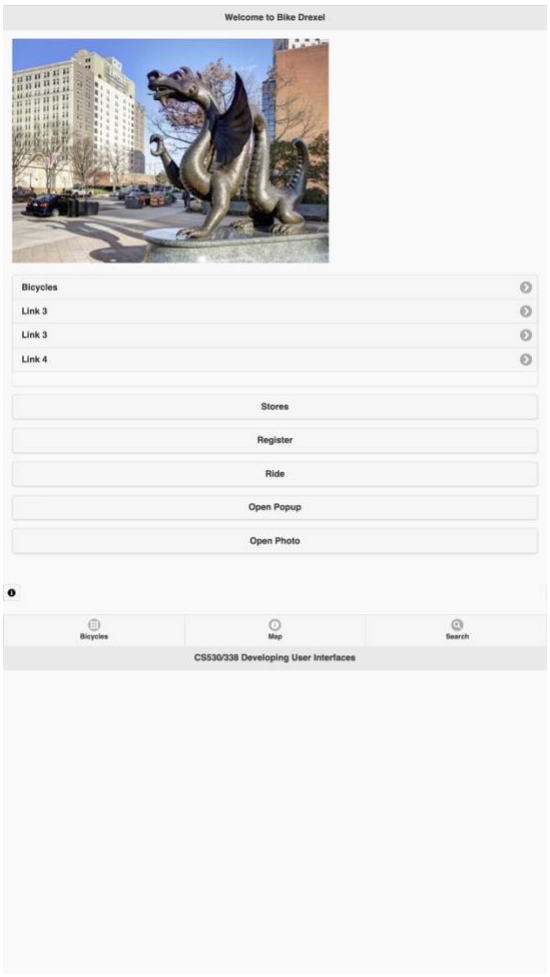
Step 18, navbar: To create a navigation bar we need to add a navbar div to the footer. Replace the footer of #welcome page with the following code:

```
<div data-role="footer">
   <div data-role="navbar">
       <ul>
           <li> <a href="#bicycles" data-icon="grid">Bicycles</a>
           <li> <a href="#map" data-icon="info">Map</a>
           <li> <a href="" data-icon="search">Search </a>
       </ul>
   </div>
   <h4>CS530/338 Developing User Interfaces </h4>
</div>
```

We added icons for every navbar element using the standard JQuery Mobile icon mechanism: data-icon with a standard name. Alternatively, we can customize the navbar using our own icons. An icon needs to be an image 18x18 pixels. Read here about using customized icons: http://demos.jquerymobile.com/1.1.2/docs/buttons/buttons-icons.html

There are plenty of free icons available on the web. Try this popular service:
http://www.glyphish.com

Add the navbar for every page where you want to have the same footer. This is how it should look like:



Collapsible Content

Collapsible content is especially useful in mobile devices where space is limited. Collapsible content can be hidden and revealed after touching a title.

To create collapsible content we need to define a `div` with `data-role="collapsible"`. This container needs to have a header element that will act as the title to be shown always as the open/close button. The collapsible content will be any HTML code inside that container apart from the title.

Step 19, collapsible content: add the following code to the `content div` of your `welcome` page:

```
<div data-role="collapsible">
      <h3>Section 1</h3>
      <p>I'm the collapsible content for section 1</p>
</div>
```

Test your application and see how JQuery Mobile renders your code.

Step 20: By default, jQuery Mobile does not open collapsible content when the page loads. We can show it by default using the attribute `data-collapsed="false"` inside the `div` container. Add this attribute and test your code.

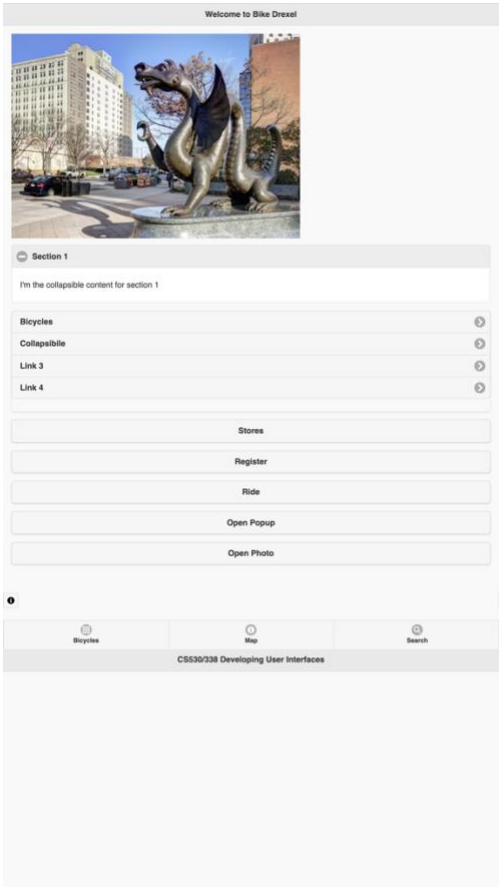
Step 21, Accordion: The Accordion behavior allows us to group collapsible content so that only one panel can be visible at a time. So, when the user views one panel, all the others are closed. To implement this behavior, we need to define a container with `data-role="collapsible-set"` with a group of `collapsible` panels as children.

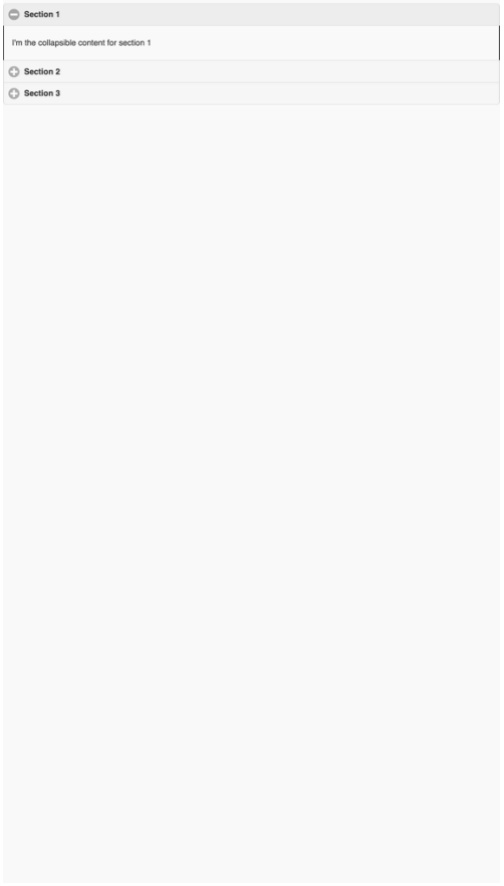
Create a new `page div` in your application and name it collapsible (e.g. `id="collapsible"`). Add the following code to the content div of your new page:

```
<div data-role="collapsible-set" data-theme="c" data-content-theme="d">
   <div data-role="collapsible">
       <h3>Section 1</h3>
       <p>I'm the collapsible content for section 1</p>
   </div>
   <div data-role="collapsible">
       <h3>Section 2</h3>
       <p>I'm the collapsible content for section 2</p>
   </div>
   <div data-role="collapsible">
       <h3>Section 3</h3>
       <p>I'm the collapsible content for section 3</p>
   </div>
</div> <!-- collapsible-set -->
```

Link to your new page from the welcome page instead of "link 2" and test your code.

Checkpoint: Your Welcome and Collapsible page at this stage should look something like this:





You are now ready to build professional looking multi-page applications using jQuery Mobile. You can add additional content to your pages and clean up any of the extra elements on your pages that were used as examples.

When you are done, make sure to submit your URL for Lab 4 in BBLearn. Note that you should adjust the sizing in your mobile app, so it looks decent on your phone.