

Developing Mobile Websites

jQuery Mobile

Lesson 1, Activity 2: jQuery Mobile

Mobile Focus

jQuery Mobile is "a unified user interface system that works seamlessly across all popular mobile device platforms, built on the rock-solid jQuery and jQuery UI foundation." With thoroughly tested code that works consistently across all major phone platforms, support for touch and other gestures, and an Ajax-based page transition model, jQuery Mobile offers an easy way to build highly mobile-optimized sites and Web applications.

jQuery Mobile is a user-interface library built on top of the popular jQuery JavaScript library. To use jQuery Mobile, you must include jQuery first before including jQuery Mobile.

jQuery Mobile makes building mobile Web sites and mobile Web applications quicker and easier, with WYSIWYG builders, code tested on lots of devices, and easy-to-add UI widgets. But one need not, of course, use jQuery Mobile to use jQuery in mobile development, and one need not use jQuery at all to develop sites optimized for mobile viewing.

Does jQuery Mobile work for all devices - desktop, tablet, and mobile? jQuery Mobile is certainly more mobile-centric than most other frameworks; you'll have to decide if jQuery's strategy - "A unified user interface system that works seamlessly across all popular *mobile* device platforms, built on the rock-solid jQuery and jQuery UI foundation" - works for your site or application on desktops as well. Sites built with jQuery Mobile will certainly work on desktops; the library has been tested on all major desktop browsers.

jQuery UI is a similar user-interface library, built on top of jQuery, aimed at desktops. Learn more at jqueryui.com.

Supported Platforms

jQuery Mobile offers support for a wide range of platforms; they categorize browsers according to A-, B-, and C-level groupings, as below.

jQuery Mobile Graded Browser Support	
Browser Grade	Functionality
A-grade	Full enhanced experience with Ajax-based animated page transitions
B-grade	Basic, nonenhanced HTML experience that is still functional
C-grade	Enhanced experience except without Ajax navigation features

See the jQuery Mobile docs for a [full list of supported devices](#).

Lesson 1, Activity 3: The Basics

The Page Model

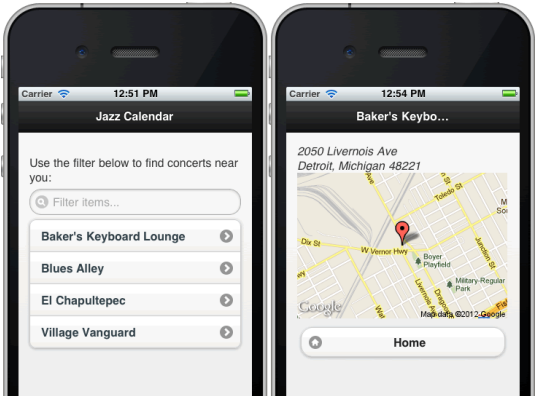
Pages in jQuery Mobile site are represented by `<div data-role="page">` with jQuery Mobile enabling Ajax-based transition between pages. You can organize pages into different files or all in the same file (with multiple `<div data-role="page">`s); regardless, jQuery Mobile will enable Ajax transitions between the different pages.

UI Elements

Lists, filters, buttons, grids - jQuery Mobile comes with a useful array of user-interface elements. Adding the `data-role="button"` attribute to a link turns it into a button; `data-role="listview"` turns an unordered list into a filter widget.

A First Example

Let's look at a simple jQuery Mobile-based demonstration: we'll rig up a low-fi page from which users can filter and select from a few jazz clubs. Tapping the club brings the user to a detail page, with map:



Open the file [jQueryMobileDemos/jazz/index.html](#) in your browser (preferably from a smartphone) and in your file editor.

Code Sample:

[jQueryMobileDemos/jazz/index.html](#)

```
Jazz Calendar

Jazz Calendar

Use the filter below to find concerts near you:


- Baker's Keyboard Lounge
- Blues Alley
- El Chapultepec
- Village Vanguard

```

We include three external files - a stylesheet and two JavaScript files, for jQuery and jQuery Mobile. These three files are all hosted externally, at [code.jquery.com](#). We could, of course, download and host these files ourselves, locally - we would need to do this if working without an internet connection.

Three divs wrap our content, with `data-roles` of "page", "header", and "content", respectively; these specific values for the `data-role` attribute are significant and specific for jQuery Mobile - they represent a page, the header for the top of the page, and the content for the page.

An unordered list with a `data-role` value of "list-view" sets up a nice-looking list-view menu; `data-filter="true"` enables the filter-as-you-type functionality, limiting the results shown to those items whose display text matches the typed text.

Tapping the link to any of the detail pages shows the title of the page, the venue's address, a map of the location, and a link ("Home") back to the main page. Let's take a look at code from one of those pages:

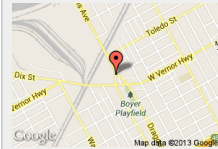
Code Sample:

[jQueryMobileDemos/jazz/bakerskeyboardlounge.html](#)

```
---- CODE OMITTED ----

Baker's Keyboard Lounge

2050 Livernois Ave
Detroit, Michigan 48221



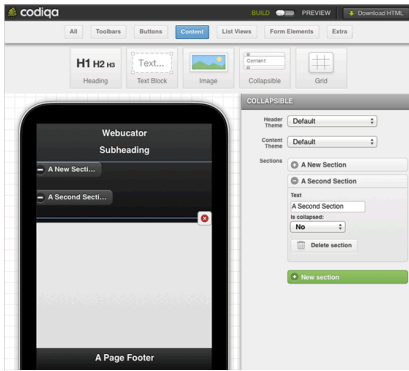
Home

---- CODE OMITTED ----
```

Again, divs with data-roles of "page", "header", and "content", respectively, mark up our content. We exploit the convenient Google Maps static-map-from-address image: supplying GET parameters for the center of the map and the marker to show allows us to easily render a useful map of the location. Setting data-role="button" for the link back to the "Home" page displays the bottom link as a mobile-friendly button; the data-icon="home" attribute for the link sets the icon for the button as jQuery Mobile's canned "home" icon.

Lesson 1, Activity 4: **Drag-and-Drop Code Builder**

A useful tool in learning to write jQuery Mobile code is the drag-and-drop UI builder (from Codiqua) right on the [jQuery Mobile home page](#). Select some options from the top menu, drag to the screen, set text or configure as needed, and voila: clicking the **Download HTML** button at upper right offers you a well-formed template making use of jQuery Mobile's canned UI elements.



Lesson 1, Activity 6: Gestures

Like all JavaScript frameworks, jQuery Mobile offers support for event handling - convenient ways to respond programmatically to user interaction with our pages. Usefully, this includes touch events - responding to finger taps and swipes to change pages, open dialogs, etc. Here are the touch events, from the [jQuery Mobile docs](#).

jQuery Mobile Touch Events	
Event	Details
tap	Triggers after a quick, complete touch event
taphold	Triggers after a held complete touch event (close to one second)
swipe	Triggers when a horizontal drag of 30px or more (and less than 20px vertically) occurs within 1 second duration (but these can be configured)
swipeleft	Triggers when a swipe event occurred moving in the left direction
swiperight	Triggers when a swipe event occurred moving in the right direction

Let's take a look at an example.

Swiping to Change Pages

We'll extend our find-a-jazz-club pages from the previous example to allow the user to swipe right and left to navigate from one detail page to the next, rather than having to return to the home screen each time. Open [jQuery Mobile Demos/jazz-touch/index.html](#) from a smartphone or emulator, and take a look at the code in your file editor.

Code Sample:

[jQuery Mobile Demos/jazz-touch/index.html](#)

```
<!DOCTYPE html>
<html>
<head>
<title>Jazz Calendar</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="http://code.jquery.com/mobile/1.1.0-rc.2/jquery.mobile-1.1.0-rc.2.min.css" />
<script src="http://code.jquery.com/jquery-1.7.1.min.js"></script>
<script src="http://code.jquery.com/mobile/1.1.0-rc.2/jquery.mobile-1.1.0-rc.2.min.js"></script>
<script type="text/javascript">
$(document).on('swipeleft', "#bakerskeyboardlounge", function() {
$.mobile.changePage("#villagevanguard");
});
$(document).on('swiperight', "#bakerskeyboardlounge", function() {
$.mobile.changePage("#bluesalley");
});
$(document).on('swipeleft', "#bluesalley", function() {
$.mobile.changePage("#bakerskeyboardlounge");
});
$(document).on('swiperight', "#bluesalley", function() {
$.mobile.changePage("#elchapultepec");
});
$(document).on('swipeleft', "#elchapultepec", function() {
$.mobile.changePage("#bluesalley");
});
$(document).on('swiperight', "#elchapultepec", function() {
$.mobile.changePage("#bluesalley");
});
$(document).on('swipeleft', "#villagevanguard", function() {
$.mobile.changePage("#elchapultepec");
});
$(document).on('swiperight', "#villagevanguard", function() {
$.mobile.changePage("#bakerskeyboardlounge");
});
</script>
</head>
<body>

<div data-role="page" id="home">
<div data-role="header">
<h1>Jazz Calendar</h1>
</div>

<div data-role="content">
<p>Use the filter below to find concerts near you:</p>
<ul data-role="listview" data-inset="true" data-filter="true">
<li><a href="#bakerskeyboardlounge">Baker's Keyboard Lounge</a></li>
<li><a href="#bluesalley">Blues Alley</a></li>

<li><a href="#elchapultepec">El Chapultepec</a></li>
<li><a href="#villagevanguard">Village Vanguard</a></li>
</ul>
</div>
</div>

<div data-role="page" id="bakerskeyboardlounge">

<div data-role="header">
<h1>Baker's Keyboard Lounge</h1>
</div>

<div data-role="content">
<address>2050 Livernois Ave
<br>Detroit, Michigan 48221</address>

<a href="index.html" data-role="button" data-icon="home">Home</a>
</div>
</div>

<div data-role="page" id="bluesalley">

<div data-role="header">
<h1>Blues Alley</h1>
</div>

<div data-role="content">
<address>1073 Wisconsin Ave. NW
<br>Washington, DC 20007</address>

<a href="index.html" data-role="button" data-icon="home">Home</a>
</div>
</div>

<div data-role="page" id="elchapultepec">

<div data-role="header">
<h1>El Chapultepec</h1>
</div>

<div data-role="content">
<address>1962 Market St
<br>Denver, CO 80202</address>

<a href="index.html" data-role="button" data-icon="home">Home</a>
</div>
</div>

<div data-role="page" id="villagevanguard">

<div data-role="header">
<h1>Village Vanguard</h1>
</div>

<div data-role="content">
<address>178 7th Avenue South, New York, NY 10014
<br></address>

<a href="index.html" data-role="button" data-icon="home">Home</a>
</div>
</div>

</div>
```

```
</body>
</html>
```

The first thing you notice is that there's only one file here. We've changed the file architecture from the last example: where the previous demo code presented each individual-venue page as a separate file, we now present each individual-venue page as separate divs, each with `data-role="page"`, in the same index.html file. Note that each page div has a unique id, which we'll use to reference the "page" in links. To the user, this is transparent - jQuery Mobile's Ajax-based page transitions hide this implementation detail.

In the head of the page, we create rules for each page's `swipeleft` and `swiperight` event: swiping right on the `#bakerskeyboardlounge`, for instance, brings the user to the `#bluesalley` page. The code `$.mobile.changePage('#bluesalley')` effects this.

One can see that this process - detailing the right- and left-swipe page links for each page - might get pretty onerous as the number of pages grows. Let's see if we can handle this a little more elegantly.

Using data Attributes

For a next iteration of the swipe-able pages, we'll use HTML5 `data-` attributes. HTML5, unlike earlier HTML versions, now formally allows - encourages, even - parameter information passing via attributes named `data-*` on any element. For our purposes here, this means each venue's div can hold information about the pages to which swiping right and left should bring the user.

```
<div data-role="page" data-next-page="#elchapultepec" data-prev-page="#bakerskeyboardlounge" id="bluesalley">
```

Using the data attributes, we can write JavaScript event-handling code that is clearer and more concise. Also, this strategy would be much easier to implement if the page for each venue were produced by a content management system.

Open [jQuery Mobile/Demos/jazz/touch/index.attribute.html](#) from a mobile device or emulator; check out the code with a file editor.

Code Sample:

[jQuery Mobile/Demos/jazz/touch/index.attribute.html](#)

```
---- C O D E   O M I T T E D ----
<script type="text/javascript">

    $(document).on('swipeleft', ".swipeablepage", function() {
        $.mobile.changePage($(this).attr('data-next-page'));
    });

    $(document).on('swiperight', ".swipeablepage", function() {
        $.mobile.changePage($(this).attr('data-prev-page'));
    });

</script>
---- C O D E   O M I T T E D ----

<div data-role="page" id="home">
<div data-role="header">
    <h1>Jazz Calendar - Attribute</h1>
</div>

<div data-role="content">
    <p>Use the filter below to find concerts near you:</p>
    <ul data-role="listview" data-inset="true" data-filter="true">
        <li><a href="#bakerskeyboardlounge">Baker's Keyboard Lounge</a></li>
        <li><a href="#bluesalley">Blues Alley</a></li>
        <li><a href="#elchapultepec">El Chapultepec</a></li>
        <li><a href="#villagevanguard">Village Vanguard</a></li>
    </ul>
</div>
</div>

<div data-role="page" data-next-page="#bluesalley" data-prev-page="#villagevanguard" class="swipeablepage" id="bakerskeyboardlounge">

<div data-role="header">
    <h1>Baker's Keyboard Lounge</h1>
</div>

<div data-role="content">
    <address>2050 Livernols Ave
    <br>Detroit, Michigan 48221</address>
    
    <a href="#home" data-role="button" data-icon="home">Home</a>
</div>
</div>

---- C O D E   O M I T T E D ----
```

We note that the JavaScript handling the swipe events is much more brief. We bind the `swipeleft` and `swiperight` to any div of class `swipeablepage` - we've chosen not to offer swiping from the home page, though of course we could.

For each swipe event, bring the user (`$.mobile.changePage`) to the page represented by the id contained in the respective div's `data-next-page` (or `-prev-page`): `$.mobile.changePage($(this).attr('data-next-page'))`

Lesson 1, Activity 8: Handling Orientation Change

Mobile phones, unlike desktops and laptops, often change orientation: we tilt our phones from vertical (portrait) to horizontal (landscape) orientation. Web pages and apps that respond to orientation changes are more useful and more interesting.

jQuery Mobile makes easy the handling of these events with the orientationchange event; see the [jQuery Mobile docs](#) for more info:

jQuery Mobile Orientation Event	
Event	Details
orientationchange	Triggers when a device orientation changes (by turning it vertically or horizontally). When bound to this event, your callback function can leverage a second argument, which contains an orientation property equal to either "portrait" or "landscape". These values are also added as classes to the HTML element, allowing you to leverage them in your CSS selectors. Note that we currently bind to the resize event when orientationchange is not natively supported, or when \$.mobile.orientationChangeEnabled is set to false.

Showing/hiding content, changing a side-by-side layout into a stacked vertical layout, and other orientation-state reactions are enhancements to your pages that your mobile users will appreciate.

Let's look at an example with the Jazz Calendar site: open [jQuery Mobile/Demos/jazz-orientation/index.html](#) in a phone browser and with a file editor to check the code.

Code Sample:

[jQuery Mobile/Demos/jazz-orientation/index.html](#)

```
---- CODE OMITTED ----

<script type="text/javascript">

$(document).on('swipeleft', '.swipeablepage', function() {
    $.mobile.changePage($(this).attr('data-next-page'));
});

$(document).on('swiperight', '.swipeablepage', function() {
    $.mobile.changePage($(this).attr('data-prev-page'));
});

$(window).bind('orientationchange', function(event) {
    setContent(event.orientation);
});

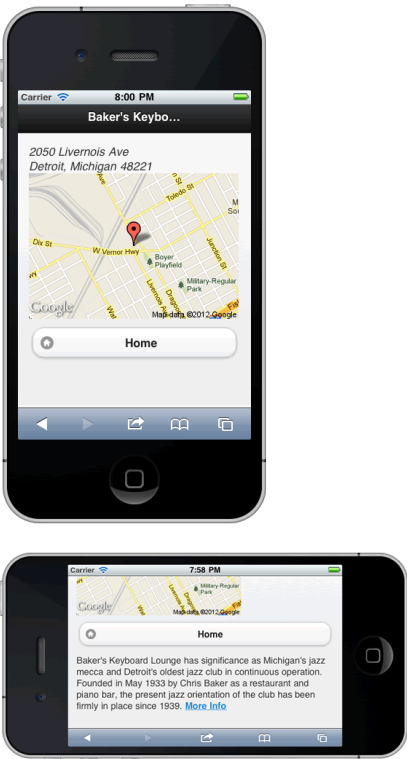
$(document).on('pageinit', '.page', function() {
    setContent(window.orientation);
});

function setContent(orien) {
    // desktops will return undefined for window.orientation, so handle that:
    if (orien != undefined) {
        // on page load, window.orientation returns 0, 90, ~90, 180, so translate to 'portrait' or 'landscape'
        if (orien == '0' || orien == '180') {
            orien = 'portrait';
        } else if (orien == '90' || orien == '~90') {
            orien = 'landscape';
        }
    }

    //hide extra content for portrait view
    if (orien == 'portrait') {
        $('p.info').hide();
    } else if (orien == 'landscape') {
        $('p.info').show();
    }
}

</script>
---- CODE OMITTED ----
```

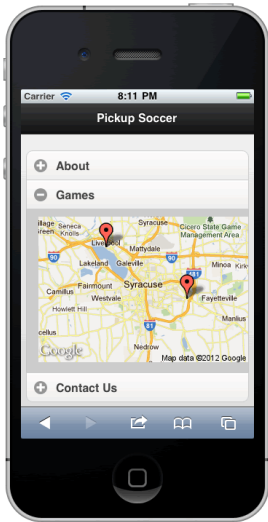
We create a function, setContent, which when executed will check to see if the parameter passed to it (orien) is undefined; if not, then we must be dealing with a device for which orientation makes sense - a phone rather than a desktop. setContent can be invoked in two different ways. When the page initially loads (\$(document).on('pageinit', '.page', function())), we pass the value of window.orientation to setContent - this has a value of 90 or -90 (for landscape orientations) or 0 or 180 (for portrait orientations). We also check for orientation-state changes (\$(window).bind('orientationchange', function(event))), passing the value of event.orientation to setContent. This parameter has a value of "landscape" or "orientation". If its parameter has one of the numeric values, function setContent translates the value into "landscape" or "portrait". It then uses jQuery's hide() and show() functions to hide or show any paragraph of class info - showing the content for landscape and hiding it for portrait:



Lesson 1, Activity 10: Pickup Soccer with jQuery Mobile

Duration: 25 to 35 minutes.

Let's use the jQuery UI builder to create a mobile-optimized page for the Pickup Soccer site. We'll add a simple header and footer and use a "collapsible-set" element to let the user pick from a few different content choices:



1. Visit the [jQuery Mobile home page](#) from your browser; scroll down to the Codiqua UI builder;
2. Drag a header and footer to the page - set appropriate text for each;
3. Drag a "Collapsible" element to the screen;
4. Add sections for "About", "Games", and "Contact Us";
5. Add appropriate content for each section - add a map, with a few markers - to the "Games" section;
6. When you are satisfied with your page, download the HTML to your computer;
7. Unzip the downloaded archive; save the file app.html to your Web accessible site;
8. View the page on a smartphone;
9. Open the file in an editor to check the code.

Solution:[jQueryMobileSolutions/soccer/app.html](#)

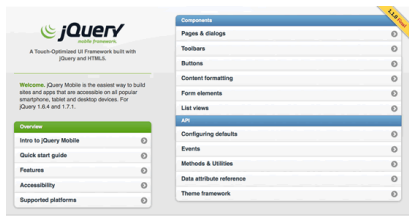
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>
    </title>
    <link rel="stylesheet" href="http://code.jquery.com/mobile/1.0.1/jquery.mobile-1.0.1.min.css" />
    <style>
      /* App custom styles */
    </style>
    <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js">
    </script>
    <script src="http://code.jquery.com/mobile/1.0.1/jquery.mobile-1.0.1.min.js">
    </script>
  </head>
  <body>
    <div data-role="page" id="page1">
      <div data-theme="a" data-role="header">
        <h3> Pickup Soccer
        </h3>
      </div>
      <div data-role="content">
        <div data-role="collapsible-set" data-theme="" data-content-theme="b">
          <div data-role="collapsible" data-collapsed="true">
            <h3> About
            </h3>
            <div>
              <b>
                Pickup Soccer is your place to find a game!
              </b>
            </div>
          </div>
          <div data-role="collapsible" data-collapsed="false">
            <h3> Games
            </h3>
            
          </div>
          <div data-role="collapsible" data-collapsed="true">
            <h3> Contact Us
            </h3>
            <div>
              123 Fake Street, Anywhere, USA
            </div>
          </div>
        </div>
      </div>
      <div data-theme="a" data-role="footer">
        <h3> Find a Game. Now.
        </h3>
      </div>
    </div>
    <script>
      //App custom javascript
    </script>
  </body>
</html>
```

Not much coding to do here - none, really! The Codiqua UI builder has done all the work for us. Of course, the WYSIWYG builder has its limitations - more advanced behaviors would require some custom coding. But a nice way to start!

Lesson 1, Activity 11: What about Desktops?

As mentioned above, jQuery Mobile is a mobile-centric library, emphasizing UI widgets and presentation specifically for smartphones and tablets. But (as also mentioned above), jQuery Mobile certainly works fine for desktop browsers. Offering a better desktop experience is easy enough with the addition of some CSS3 media queries.

Consider the jQuery Mobile [main documentation page](#). When viewed on a desktop at a browser width greater than 650 pixels, the main navigation element sits to the left of the main content:



At browser widths narrower than 650 pixels, the design moves to a one-column layout, with the nav on top of the main content:



The site accomplishes this with a custom stylesheet. A custom CSS stylesheet - that is, a CSS file the authors of the site have added themselves, separate from the standard jQuery Mobile stylesheet - directs the div layer holding the nav to float left when the browser is wider than 650 pixels:

```
@media all and (min-width: 650px) {
  /* some code omitted */
  .content-secondary {
    text-align: left;
    float: left;
    width: 45%;
    background: none;
  }
  /* some code omitted */
}
```

As designers and developers, we can add whatever custom styles we want to our jQuery Mobile sites, enhancing the user experience for visitors who view our sites and Web applications on desktop browsers.