

# jQuery Mobile

What Is jQuery Mobile?

## Lesson 1, Activity 2: Introducing jQuery Mobile

We as web developers and designers have always wrestled with the challenge of meeting the diverse needs of users who view our work on many different devices. Internet Explorer might render a page differently from Firefox or Opera - and don't forget to account for older versions of the same browser. Connection speeds, screen resolutions, operating systems, use of screen readers - our code must work for all cases.

The explosive proliferation of smartphones and tablets over the past few years has added to our challenge. Screens are smaller, to be sure, and the differences between devices are greater than ever - and those devices come with new capabilities we didn't see on desktops and laptops. Wouldn't it be nice if the code we write for a given web page worked for a new Kindle Fire, an iPad, and a Windows Phone? And if we could exploit the device's orientation, GPS, and accelerometer?

jQuery Mobile is a tool to address these challenge. It is a user-interface framework, built on top of the jQuery JavaScript framework. jQuery Mobile offers a "unified user interface system that works seamlessly across all popular mobile device platforms, built on the rock-solid jQuery and jQuery UI foundation." For us, that means we can write code that will work consistently for lots of devices, address the unique challenges of smartphones and tablets, and leverage new device capabilities.

For this course, we will be working with jQuery Mobile version 1.1, the current stable version. Please note that future versions may change some of the API.

### What jQuery Is Not

Keep in mind that jQuery Mobile isn't "jQuery for mobile". It isn't a replacement for the jQuery library for mobile use; rather, it is a user-interface library built on top of jQuery. You must include jQuery, as well as the jQuery Mobile stylesheet, to exploit its capabilities.

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 it works best 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 for desktops, also built on top of jQuery. Learn more at [jqueryui.com](http://jqueryui.com).

jQuery Mobile also isn't an SDK to leverage native phone capabilities. While writing native phone apps with jQuery Mobile is possible (with PhoneGap, which we'll look at later), it takes some extra work.

### 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. A-level browsers can exploit full functionality (though, as [the docs](#) note, the "visual fidelity of the experience and smoothness of page transitions are highly dependent on the CSS rendering capabilities of the device and platform so not all A grade experience will be pixel-perfect but that's the nature of the web." See the jQuery Mobile docs for the current [full list of supported devices](#).

**jQuery Mobile Graded Browser Support**

Browser Grade	Details
A-grade	Full enhanced experience with Ajax-based animated page transitions
B-grade	Basic, non-enhanced HTML experience that is still functional
C-grade	Enhanced experience except without Ajax navigation features

### Progressive Enhancement

Progressive Enhancement is a web-design strategy in which newer devices/browsers/capabilities receive an enhanced version of our pages, while less capable devices/browsers still receive basic content and functionality. Progressive Enhancement emphasizes semantic markup, controlling presentation with external stylesheets, and use of JavaScript to add functionality.

As [the docs](#) state, jQuery Mobile's "[p]rogressive enhancement approach brings core content and functionality to all mobile, tablet and desktop platforms and a rich, installed application-like experience on newer mobile platforms." As the A/B/C chart above suggests, high-functioning devices and browsers get the good stuff - Ajax page transitions, native-app-like functionality, the "full enhanced experience" - while older or less capable devices and browsers still work, to the degree available. That's a huge help to us as developers: we can exploit new features and cool device capabilities without alienating users with older, less capable devices.

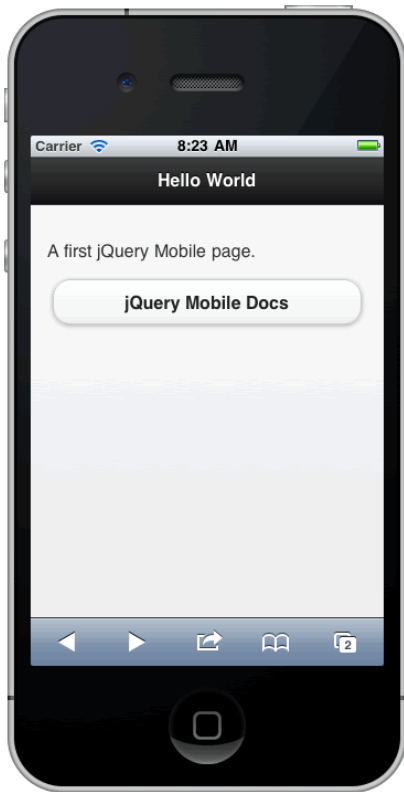
### Accessibility Support

Web accessibility refers to the process of making websites useable for people who employ screen readers and other assistive technologies. A person with limited or no sight might, for example, uses the iPhone's [VoiceOver](#) to read aloud the contents of a web page.

With its focus on standard, semantic markup, jQuery Mobile provides a solid foundation for creating accessible sites. For newer browsers, many jQuery Mobile components "leverage techniques such as focus management, keyboard navigation, and HTML attributes" specified by accessibility standards; see the [jQuery Mobile docs](#) for more information.

### A First Example

Let's take a first look at some code. When viewed on a smartphone, the page [WhatIsjQueryMobile/Demos/helloworld/index.html](http://WhatIsjQueryMobile/Demos/helloworld/index.html) looks something like this:



Open the page in a desktop browser and with a smartphone or emulator. Open up the page in a file editor, too, to check out the code.

#### Code Sample:

[WhatIsjQueryMobile/Demos/helloworld/index.html](http://WhatIsjQueryMobile/Demos/helloworld/index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>jQuery Mobile: Hello World</title>
    <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>
  </head>
  <body>
    <!-- the "page" div is the basic unit of jQuery Mobile - a page for the user -->
    <div data-role="page">
      <!-- this is a header, appearing at the top of the page -->
      <div data-role="header" data-position="inline">
        <h1>Hello World</h1>
      </div>
      <div data-role="content">
        <p>A first jQuery Mobile page.</p>
        <!-- this is a button -->
        <a href="http://jquerymobile.com/demos/1.1.0-rc.2/" data-role="button">jQuery Mobile Docs</a>
      </div>
    </div>
  </body>
</html>
```

While we'll get into lots more detail about code later in this course, we can point out a few things now. First, note that the page is a valid HTML5 document. The `DOCTYPE` is `html`. The `div`s on the page make use of the HTML5 data attribute. The outermost `div`, for instance, has attribute `data-role="page"`, which is significant for jQuery Mobile; we'll learn later in the course about `div`s as pages.

Second, note that the page links to some external resources - the two JavaScript files and one stylesheet are hosted at `code.jquery.com`, with nothing other than "index.html" on our end.

Let's check jQuery Mobile's progressive enhancement features. Consider the following two screenshots, each showing [WhatIsjQueryMobile/Demos/helloworld/index.html](http://WhatIsjQueryMobile/Demos/helloworld/index.html) in Firefox:

WhatIsjQueryMobile/Demos/helloworld/index.html with Javascript on:



WhatIsjQueryMobile/Demos/helloworld/index.html with Javascript off:

## Hello World

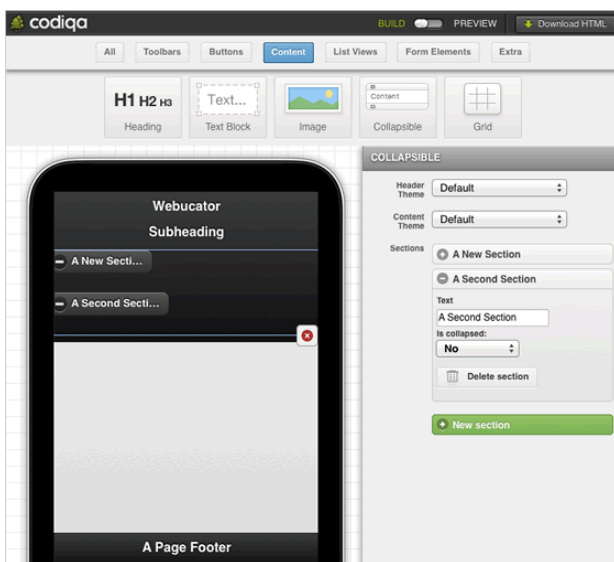
A first jQuery Mobile page.

[jQuery Mobile Docs](#)

The top screenshot, with JavaScript on, shows a page similar to the smartphone view. The bottom screenshot shows that the page still works without JavaScript enabled: we don't get the nice styling, but the content is visible and the link works for users who don't have, or who have turned off, JavaScript.

### jQuery Mobile WYSIWYG Builders

jQuery Mobile offers several what-you-see-is-what-you-get UI tools, allowing you to create pages from a drag-and-drop interface and, when finished, to download the resulting code. Visit the [Codiqa UI builder](#) on the jQuery Mobile website from a desktop browser:



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.

### HTML5

HTML5 is the latest revision of the web markup language standard - it addresses the shortcomings of HTML 4 and XHTML, and also adds new features. Adoption of HTML5 depends, of course, on the degree to which browsers support HTML5 features. Luckily for us, support - especially for the features we'll focus on here - is especially strong among mobile browsers.

HTML5 is in part an acceptance of the fact that browsers tolerate lots of bad code, that perhaps we need not be so strict with requiring end tags and case sensitivity, and that making obsolete some ten years' worth of existing web pages would be catastrophic.

As an example of this flexibility, all of the following are permitted in HTML5:

1. `<link type="text/css" href="style.css" />`
2. `<LINK TYPE="text/css" HREF="style.css" />`
3. `<link type=text/css href=style.css>`
4. `<LINK TYPE=text/css HREF=style.css>`
5. `<LiNk TyPe=text/css hReF="style.css">`

As the above shows:

1. HTML5 is case insensitive.
2. HTML5 allows for unclosed tags, but you can use the XML-style shortcut close tag if you want, as shown in the first two examples above.
3. HTML5 does not require quotes around attribute values unless the values have spaces in them - `<div class="news featured">`, for example.

This new flexibility could lead to a bit of chaos on your development team. Different HTML authors will take different approaches. Our recommendation is that you choose one approach and stick to it. In this course, for example, we use the following guidelines:

1. Write tags and attributes in all lowercase letters (even event handlers like `onclick`).
2. Do not use shortcut close tags for void/empty elements.
3. Put all attribute values in quotes. (Why? Because attribute values that have spaces in them have to be in quotes. And I do not like the idea of having some attributes in quotes and some not.)
4. Minimize attributes when you can.

Of course, your choices here must also take into account your expected audience's use of older browsers - a self-closed `div`, for instance, won't work in Internet Explorer 7 and 8.

## New HTML5 Features

The table below shows the new elements that HTML5 has introduced. Note that we won't cover most of these in this course.

New HTML5 Elements	
Tag	Description
<code>&lt;article&gt;</code>	For a standalone article on a page. Articles can be nested within other articles; for example, a blog post would be contained in <code>&lt;article&gt;</code> tags and each comment would be contained within a nested <code>&lt;article&gt;</code> tag.
<code>&lt;aside&gt;</code>	For content contained in an aside. Often used for navigation elements or for a list of articles or categories (e.g., in a blog).
<code>&lt;audio&gt;</code>	For embedding audio files.
<code>&lt;canvas&gt;</code>	For creating drawings natively in the browser.
<code>&lt;command&gt;</code>	For command buttons similar to what you might see in the Microsoft Office 2010 ribbon. <code>&lt;command&gt;</code> must be nested in <code>&lt;menu&gt;</code> .
<code>&lt;datalist&gt;</code>	For a drop-down list providing built-in functionality similar to a JavaScript autocomplete boxes.
<code>&lt;details&gt;</code>	For expandable (usually initially hidden) content to provide more information about an element.
<code>&lt;embed&gt;</code>	For backwards compatibility with the nonstandard (but well supported) <code>&lt;embed&gt;</code> tag in HTML 4.
<code>&lt;figcaption&gt;</code>	For captions on images. (In HTML 4, there was no way to semantically associate a caption with an image.
<code>&lt;figure&gt;</code>	For wrapping embedded content (e.g., an image) with a <code>&lt;figcaption&gt;</code> .
<code>&lt;footer&gt;</code>	For the footer of a page or a section.
<code>&lt;header&gt;</code>	For the header of a page or a section.
<code>&lt;hgroup&gt;</code>	For grouping <code>&lt;h1&gt;...&lt;h6&gt;</code> tags on a page. For example, the title and subtitle of a page could be an <code>&lt;h1&gt;</code> and <code>&lt;h2&gt;</code> grouped in an <code>&lt;hgroup&gt;</code> tag.
<code>&lt;keygen&gt;</code>	For a generated key in a form.
<code>&lt;mark&gt;</code>	For showing marked (or highlighted) text. Unlike <code>&lt;strong&gt;</code> or <code>&lt;em&gt;</code> , <code>&lt;mark&gt;</code> doesn't give the text any special meaning. The best example is marking a word or phrase that a user has searched on within the search results.
<code>&lt;meter&gt;</code>	For a measurement within a set range.
<code>&lt;nav&gt;</code>	For holding a group of navigation links.
<code>&lt;output&gt;</code>	For holding output (e.g., produced by a script).
<code>&lt;progress&gt;</code>	For a progress indicator (e.g., for a loading graphic).
<code>&lt;rp&gt;</code>	Used within <code>&lt;ruby&gt;</code> tags to tell browsers that cannot render the East Asia characters properly what extra characters (usually parentheses) to display.
<code>&lt;rt&gt;</code>	Used within <code>&lt;ruby&gt;</code> tags to show how to pronounce East Asia characters.
<code>&lt;ruby&gt;</code>	For ruby annotations. (See <a href="http://www.w3.org/TR/ruby">http://www.w3.org/TR/ruby</a> for examples.)
<code>&lt;section&gt;</code>	For creating a <code>&lt;section&gt;</code> on the page. This helps the browser (user agent) determine the page outline.
<code>&lt;source&gt;</code>	For indicating media sources within <code>&lt;video&gt;</code> and <code>&lt;audio&gt;</code> .
<code>&lt;summary&gt;</code>	For the header of a <code>&lt;detail&gt;</code> element. The <code>&lt;summary&gt;</code> would show by default.
<code>&lt;time&gt;</code>	For holding a machine-readable date and/or time while displaying a friendly date and/or time.
<code>&lt;video&gt;</code>	For embedding video files.
<code>&lt;wbr&gt;</code>	An opportunity to insert a line break within a word. (e.g., super <code>&lt;wbr&gt;</code> califragilistic <code>&lt;wbr&gt;</code> expialidocious)

Keep in mind that jQuery Mobile doesn't inherently make use of all - even most - of these new tags; in fact, one might argue that jQuery Mobile could be a *little more* semantic, making use of the `<header>` or `<footer>` tag, say, instead of lots of `<div>`s. Nonetheless, the jQuery Mobile is syntactically valid HTML5.

## A First Example

Let's take a look at a quick first example - open up [WhatIsjQueryMobile/Demos/html5/index.html](http://WhatIsjQueryMobile/Demos/html5/index.html) in a browser and in your file editor to inspect the code. From a desktop browser you'll see something like this:

# Site Title

- [link 1](#)
- [link 2](#)
- [link 3](#)

## Page Title

### Subtitle

Content content content

Side bar info goes here

Footer content here

#### Code Sample:

---

WhatIsjQueryMobile/Demos/html5/index.html

```
<!doctype html>
<html>
<head>
  <title>A Simple Document</title>
</head>
<body>
  <header>
    <h1>Site Title</h1>
    <nav>
      <ul>
        <li><a href="#">link 1</a></li>
        <li><a href="#">link 2</a></li>
        <li><a href="#">link 3</a></li>
      </ul>
    </nav>
  </header>

  <section>
    <article>
      <hgroup>
        <h1>Page Title</h1>
        <h2>Subtitle</h2>
      </hgroup>
      <p>Content content content</p>
    </article>
  </section>
  <aside>
    <p>Sidebar info goes here</p>
  </aside>
  <footer>Footer content here</footer>
</body>
</html>
```

Presentationally, the page is less than stunning when viewed in a browser - we've not styled the elements at all - but the interesting bits are how we've marked up the content. Firstly, note that we've declared a `doctype` of `html` - this fact alone means this page is an HTML5 document.

Second, note that we have made use of the HTML5 semantic tags to imply meaning to the content we markup for viewing on the web:

1. A `header` tag wraps the header of the page.
2. The main navigation items are wrapped with a `nav` tag.
3. An `article` tag wraps the main content, while an `aside` tag wraps the sidebar content.

#### HTML5 Forms

Our mobile sites will need to accept user input - registration forms, comment forms, and the like to ask users to send us, via form fields, some information. jQuery Mobile handles a good bit of this, but it's good to know about how HTML5 makes available a set of new form input types and attributes that enhance the user experience. Keep in mind that adoption for some of these new features is spotty at best; we'll cover the better implemented options.

There are 13 new input types:

1. `search`
2. `tel`
3. `url`
4. `email`
5. `datetime`
6. `date`
7. `month`
8. `week`
9. `time`
10. `datetime-local`
11. `number`

We'll concentrate on a few of the more useful types for mobile.

## Telephone

The `tel` type input expects a phone number. Many phones will present a numeric keyboard:



## Dates

The `date` type of input allows the user to enter a date with no time zone. Some smartphone browsers present a date picker when a field of type `date` receives focus. iPhones with iOS version 5 or later, for instance, offer a date picker - earlier versions do not.



## Email

Fields of type `email` expect a valid email address. Many phones will show a contextually appropriate set of characters for input; the iPhone, for instance, displays the "@" character:



Some browsers will validate the field as a valid email address.

### Web Address

Fields of type `url` are for entering full website address. Many phones will show a contextually appropriate set of characters for input; the iPhone, for instance, displays the ".com" key:



Some browsers will validate the field as a valid web address.

### New Field Attributes

The `placeholder` attribute is among the more useful new HTML5 field attributes: it allows us to specify an example value for the field, a value which disappears when the user begins to



enter content in the field.

The required attribute is very useful - but, sadly, not well-supported among mobile browsers: the automatic client-side HTML5 form validation that works on desktop browsers is lacking on their mobile counterparts.

### HTML5 Form Example

Open [WhatIsiQueryMobile/Demos/forms/index.html](http://WhatIsiQueryMobile/Demos/forms/index.html) in a mobile browser - this is the page from which the above screenshots were taken.

#### Code Sample:

[WhatIsiQueryMobile/Demos/forms/index.html](http://WhatIsiQueryMobile/Demos/forms/index.html)

```

---- CODE OMITTED ----

<form action="forms.html" method="post">
  <label for="name">Name</label><br>
  <input type="text" name="name" id="name" placeholder="Jane Doe"><br>
  <br>
  <label for="phone">Phone</label><br>
  <input type="tel" name="phone" id="phone"><br>
  <br>
  <label for="email">Email</label><br>
  <input type="email" name="email" id="email"><br>
  <br>
  <label for="website">Website URL</label><br>
  <input type="url" name="website" id="website"><br>
  <br>
  <label for="DOB">Date of Birth</label><br>
  <input type="date" name="DOB" id="DOB"><br>
  <br>
  <input type="submit">
</form>
---- CODE OMITTED ----

```

We use a placeholder value for the first ("name") field, and ask the user for information of type tel, email, url, and date.

### CSS3 Media Queries

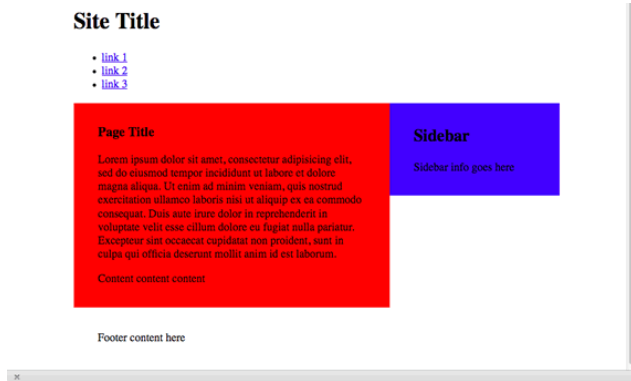
Among the new features offered as part of CSS3, media queries are easily the most useful to us as designers and developers of mobile sites. A way to apply CSS rules selectively based on both the type of media and the physical properties of the device (browser, phone) being used to access the page, each media query comprises a media type (e.g. "screen") and zero or more logical expressions - a condition evaluating to true or false based on the conditions of particular media features. We can test our user's device for screen width, device width, orientation ("portrait" or "landscape"), and other features.

**Media Query Features**

Feature	Possible Values	Min/Max?	Explanation
color	int	yes	bits per color component
color-index	int	yes	number of entries in color lookup table
device-aspect-ratio	int/int	yes	aspect ratio
device-height	length (pixels)	yes	height of the output device
device-width	length (pixels)	yes	width of the output device
grid	int	no	true for a grid-based device
height	length (pixels)	yes	height of the rendering surface
monochrome	int	yes	bits per pixel in a monochrome frame buffer
resolution	"dpi" or "dpcm"	yes	resolution
scan	"progressive" or "interlaced"	no	scanning process of "tv" media types
width	length (pixels)	yes	width of the rendering surface

Let's look at a simple example of media queries. Open [WhatIsjQueryMobile/Demos/mediaquery/index.html](http://WhatIsjQueryMobile/Demos/mediaquery/index.html) in a desktop browser; open up the same file and [WhatIsjQueryMobile/Demos/mediaquery/style.css](http://WhatIsjQueryMobile/Demos/mediaquery/style.css) in a file editor to check out the code.

When the browser is wider than 768 pixels, the page renders with the two columns floated left and right. The red main column (an article inside of a section) occupies about two-thirds of the page, at left. The blue right sidebar column (an aside) sits to the right.



When we drag the browser to a width narrower than 768 pixels, the design changes:

1. Both columns fill the width of the container element, instead of floating left and right;
2. The main column's background changes to gray;
3. The background of the sidebar column, now at bottom, changes to green.

Here's a screenshot of the narrow-browser rendering:



#### Code Sample:

[WhatIsjQueryMobile/Demos/mediaquery/index.html](http://WhatIsjQueryMobile/Demos/mediaquery/index.html)

```

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

<header>
  <h1>Site Title</h1>
  <nav>
    <ul>
      <li><a href="#">link 1</a></li>
      <li><a href="#">link 2</a></li>
      <li><a href="#">link 3</a></li>
    </ul>
  </nav>
</header>
<section>
  <article>
    <hgroup>
      <h1>Page Title</h1>
      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
        magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea
        commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
        Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>
    </hgroup>
    <p>Content content content</p>
  </article>
</section>
<aside>
  <h2>Sidebar</h2>

```

```
<p>Sidebar info goes here</p>
</aside>
---- C O D E   O M I T T E D ----
```

The HTML page contains an `article` wrapped by a `section`; an `aside` wraps the sidebar content.

### Code Sample:

[WhatIsjQueryMobile/Demos/mediaquery/style.css](#)

```
#container {
  width:80%;
  margin:0 auto;
}
section {
  float:left;
  width:55%;
  padding:2% 5%;
  background:#f00;
}
aside {
  float:right;
  width:25%;
  padding:2% 5%;
  background:#00f;
}

@media screen and (max-width: 768px) {
  section {
    float:none;
    width:90%;
    padding:5%;
    background:#ccc;
  }
  aside {
    float:none;
    width:90%;
    padding:5%;
    background:#0f0;
  }
}

footer {
  padding:5%;
  clear: both;
}
```

Unlike our earlier example, we employ a CSS stylesheet in this example. The media query dictates that "for viewing on a screen and at a maximum browser width of 768 pixels" the page should:

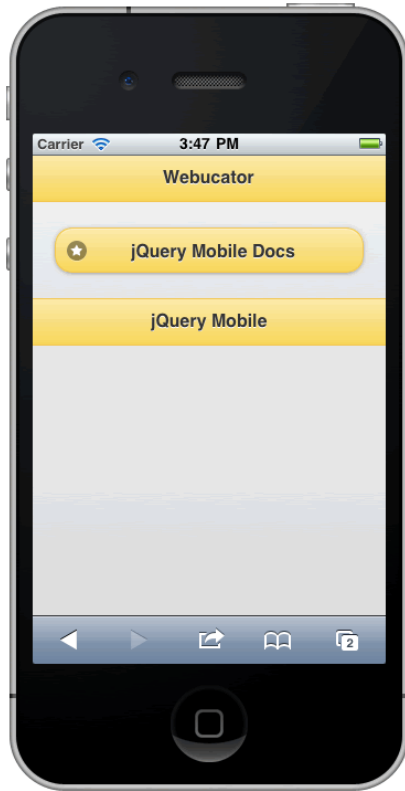
1. Unfloat the `section` element;
2. Set the width of the `section` element to 90%;
3. Set the background color of the `section` to gray;
4. Unfloat the `aside` element;
5. Set the width of the `aside` element to 90%;
6. Set the background color of the `aside` to green.

## Lesson 1, Activity 5: Building Your First jQuery Mobile Page

Duration: 10 to 20 minutes.

In this exercise, you will build a simple page using the Codiqua UI builder.

1. Visit the [Codiqua UI builder](#) from a desktop browser.
2. Drag a Page Header to the top of the screen; use the "Yellow" ("e") theme for it.
3. Drag a Page Footer to the bottom of the screen; use the "Yellow" ("e") theme for it.
4. Drag a button to the middle of the screen; set its text to "jQuery Mobile Docs"; use the "Yellow" ("e") theme for it; change the icon to "Star".
5. Try out a few other elements by dragging to the screen and configuring.
6. When you are satisfied with the page you have created, click "Download HTML" to download the code; extract "app.html" from the downloaded zip file, which will be named something like "codiqua-app-123456789.zip".
7. Move app.html to [WhatIsjQueryMobile/Exercises/](#); the directory will be empty before you add the file to it.
8. Open the new file, [WhatIsjQueryMobile/Exercises/app.html](#), in a file editor; change the button link to point to <http://jquerymobile.com/demos/1.1.0-rc.2/>
9. The page should look something like this when viewed on a smartphone:



### Solution:

[WhatIsjQueryMobile/Solutions/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 - here is where we might put our own CSS if we want to make custom style changes */
    </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>
    <!-- the WYSIWYG UI builder has created a page for us: -->
    <div data-role="page" id="page1">
      <!-- this is the header, appearing at the top of the page: -->
      <div data-theme="e" data-role="header">
        <h3>
          Webucator
        </h3>
      </div>
      <!-- this is the main content, appearing at the middle of the page, with button with the "star" icon: -->
      <div data-role="content">
        <a data-role="button" data-transition="fade" data-theme="e" href="http://jquerymobile.com/demos/1.1.0-rc.2/" data-icon="star" data-iconpos="left">
          jQuery Mobile Docs
        </a>
      </div>
      <!-- this is the footer, appearing below the content: -->
      <div data-theme="e" data-role="footer">
```

```
<h3>
  jQuery Mobile
</h3>
</div>
<script>
  //here we could write custom JavaScript
</script>
</body>
</html>
```

Other than changing the URL for the button's `href` attribute, we've written no code here. While the WYSIWYG tools won't solve all of our coding challenges, they are often a great starting point when beginning jQuery Mobile projects.