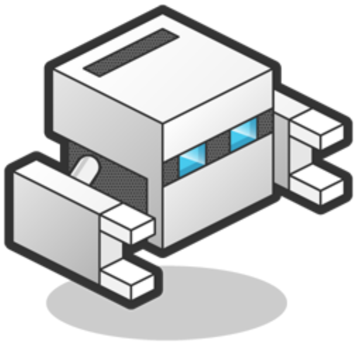


Creating Mobile Applications with jQuery Mobile and PhoneGap Build

L8507

Adobe MAX 2013
Chris Griffith

About Chris Griffith

Chris Griffith is a Staff Engineer in Qualcomm's User Experience group, and an instructor (Mobile Application Development) at UCSD Extension. Chris has over 15 years experience in developing prototypes for a variety of clients and platforms, and has developed several iPhone and Android applications. He is an Adobe Community Professional and is regularly invited to speak at conferences. In addition, he has served as a technical reviewer for several O'Reilly publications, and written for uxmag.com and the Adobe Developer Connection. In his spare time, Chris spends time with his family and sea kayaking. You can follow him on Twitter @chrisgriffith or online at chrisgriffith.wordpress.com.

[NOTE: Thoughts expressed here are my own.]

Setup and Configuration

Before you proceed, please note the following: throughout this document, the following styles are used to simplify instructions:

Menu items and field prompts are formatted in **bold**.

Text you need to enter is always in a `monospaced` font.

The ellipsis (…) indicated additional code not shown.

Required Resources

This hands-on lab required that you have the following installed (this should be on the lab systems):

Google Chrome
An HTML editor (Brackets, Edge Code, Dreamweaver, Sublime Text, etc.)
Files for this lab (zip file)
Have a PhoneGap Build account

The zip file should be expanded on your desktop. This will create a folder named L8507_assets, which will contain other files and folders.

Directory Structure

css/
Contains the css files that will be used for the lab

img/
Contains the image files that will be used for the lab

js/
Contains the Javascript files that will be used for the lab

Lab [X]/start/
Contains the starting HTML file for that lab

Lab [X]/finish/
Contains the completed HTML file for that lab

Lab 1: Creating a jQuery Mobile Template

Goal: This lab will introduce you to the basic structure of a jQuery Mobile page.

Open the file for this lab.

1. Go to your HTML editor
2. Open File Menu and select Open…
3. Open the file *Lab 1/start/index.html*
4. After the `<body>` tag add the following HTML code:

```
<section id="homePage" >
  <header>
      <h1>Page Title</h1>
  </header>
  <div>
      <p>My page's content</p>
  </div>
  <footer>
      <h4>Page Footer</h4>
  </footer>
</section>
```

5. Save your file

Test Your Code

1. Launch the Google Chrome Browser
2. Open the File menu and select Open File
3. Open the file *Lab 1/start/index.html*

Return to your html editor

1. Add `data-roles` to the section, header, div, and footer tags

```
<section id="homePage" data-role="page">
  <header data-role="header">
      <h1>Page Title</h1>
  </header>
  <div data-role="content">
      <p>My page's content</p>
  </div>
  <footer data-role="footer">
      <h4>Page Footer</h4>
```

```
    </footer>
  </section>
```

2. Save the file
3. Reload the saved file in the browser

Add a new jQuery Mobile ‘page’

1. After the </section> tag, add this code:

```
<section id="about" data-role="page" >
  <header data-role="header">
      <h1>About This App</h1>
  </header>
  <div data-role="content">
      <p>This app rocks! <a href="#homePage">Go
home</a></p>
  </div>
</section>
```

2. Now, we need to link these first ‘page’ to the second page, by adding a simple hyperlink in the first page’s content section:

```
<div data-role="content">
  <p>My page's content</p>
  <p><a href="#about">About this app</a></p>
</div>
```

3. Save the file, and preview it in the browser.

Lab 2: Buttons

Goal: Adding mobile/touch friendly buttons to the interface.

Open the file for this lab.

1. Go to your HTML editor
2. Open File Menu and select Open…
3. Open the file *Lab 2/start/index.html*
4. Add data-role=”button” to the `<a>` tag

   ```
   <p><a href="#about" data-role="button" >About this
   app</a></p>
   ```

5. Save this file, then preview it in your browser.
6. Now add data-icon=”info” to the `<a>`

   ```
   <p><a href="#about" data-role="button" data-
   icon="info">About this app</a></p>
   ```

7. Save the file, and preview it in the browser.
8. Other available icons like: grid, star, home, gear, alert.

Now let’s add a button control group.

1. In the About section, add a new `<div>` with the data-role of ‘**controlgroup’**, then we will add 3 <a> tags.

   ```
   <section id="about" data-role="page" >
     <header data-role="header">
          <h1>About This App</h1>
     </header>
     <div data-role="content">
          <p>Does this app rocks?</p>
          <div data-role="controlgroup" data-type="horizontal">
               <a href="#homePage" data-role="button">Yes</a>
               <a href="#homePage" data-role="button">No</a>
               <a href="#homePage" data-role="button">Maybe</a>
          </div>
     </div>
   </section>
   ```

2. Save the file, and preview it in the browser.

Lab 3: Transitions

Goal: To learn about various transitions available

Open the file for this lab.

1. Go to your HTML editor
2. Open File Menu and select Open…
3. *Open the file Lab 3/start/index.html*
4. Add a data-transition attribute to `<a>`

```
<a href="#about" data-transition="flip" data-
   role="button" data-icon="info">About this app</a>
```

5. Save the file, and preview it in the browser.

The available transitions are:

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

By default, the fade transition is used.

Note: You should test each transition on your target devices.

Lab 4: Headers

Goal: To learn how to add headers to the page.

Open the file for this lab.

1. Go to your HTML editor
2. Open File Menu and select Open…
3. Open the file *Lab 4/start/index.html*
4. For each `<header>`, add `data-position="fixed"` attribute.

   `<header` **`data-role="header" data-position="fixed"`** `>`

5. Save this file, then preview it in the browser.
6. For each `<header>`, add `data-id="appHeader"`

   `<header data-role="header" data-position="fixed"`
   `data-id="appHeader"``>`

7. Save this file, then preview it in the browser

Adding buttons to the Header

1. Let's move the About button to the header

   `<a href="#about" data-role="button" data-transition="`**`fade`**`" data-icon="info"` **`class="ui-btn-right"`**`>About</a>`

2. Delete this from the page:

   ```
   <p><a href="#about" data-role="button" data-
   transition="flip" data-icon="info">About this
   app</a></p>
   ```

3. Save this file, then preview in the browser.
4. Now, let's hide the text from our About button by adding

   `data-iconpos="notext"` to the `<a>` tag.

5. Save the file, and preview it in the browser

Lab 5: Footers

Goal: Learn about adding a footer element to the page.

Open the file for this lab.

1. Go to your HTML editor
2. Open File Menu and select Open…
3. Open the file *Lab 5/start/index.html*
4. Examine the content of the html. Note there are now three sections: **parkNamePage**, **stateListPage** and the **about** section.
5. After the content `<div>` in the **parkNamePage** section, we are going to be adding a footer element

   ```
   <footer data-role="footer" data-id="navFooter"
   data-position="fixed">
   </footer>
   ```

6. Inside this `<footer>` element, we are going to be adding a jQuery Mobile navbar.

   ```
   <div data-role="navbar" class="nav-footer">
   </div>
   ```

7. Within this `<div>`, we are going to add an `<ul>`, with each `<li>` functioning as our tab.

   ```
   <ul>
    <li><a href="#parkNamePage" id="parkNameTab" class="ui-btn-
    active ui-state-persist" >By Name</a></li>
    <li><a href="#stateListPage" id="stateTab">By State</a></li>
   </ul>
   ```

8. To set the active state of an item in a persistent toolbar, add `class="ui-btn-active ui-state-persist"` to the corresponding anchor.

Applying the footer to other pages

1. Copy the entire `<footer>` block and add it the **stateListPage** section
2. Move the `class="ui-btn-active ui-state-persist"` from the first `<a>` to the second `<a>` tag.
3. Save the file, and preview it in the browser.

Lab 6: Lists

Goal: Learn how jQuery Mobile display lists in a mobile friendly fashion

Open the file for this lab.

1. Go to your HTML editor
2. Open File Menu and select Open…
3. Open the file *Lab 6/start/index.html*
4. Within the `<div data-role="content">` for the **parkNamePage** section add this code:

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

5. Now open the *parklist.txt* file, and copy the text
6. Paste this text between the `<ul>` and the `</ul>`
7. Save the file, and preview it in the browser.

Adding list auto-dividers

1. In the `<ul>` tag add `data-autodividers="true"`

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

2. Save the file, and preview it in the browser.

Adding list filters

1. Now also add this code with the `<ul>` tag

   ```
   <ul data-role="listview" data-autodividers="true"
   data-filter="true">
   ```

2. Save the file, and preview it in the browser.
3. Within the `<div data-role="content">` for the **stateListPage** section add this code:

   ```
   <ul data-role="listview" data-filter="true" >
   </ul>
   ```

4. Now open *stateList.txt* file and copy the text
5. Paste this text between the `<ul>` and the `</ul>`

6. Save the file, and preview it in the browser.

Lab 7: Events

Goal: To understand some of the basic events that jQuery Mobile uses.

Open the file for this lab.

1. Go to your HTML editor
2. Open File Menu and select Open…
3. Open the file *Lab 7/start/index.html*
4. Add the following script tag within the `<head>` section

```
<script src="../../js/events.js"></script>
```

5. Open the *event.js* file in the js directory
6. Review the following Javascript functions

```
$(document).ready(function() {
     $('#parkNamePage').on('pageinit', function(event) {
          console.log("pageinit event");
     });

     $('#parkNamePage').on('pageshow', function(event) {
          console.log("pageshow event");
     });

     $('#parkDetailsPage').on('pagebeforeshow',
     function(event) {
          console.log("pagebeforeshow event");
     });
});
```

7. Save the file, and preview it in the browser
8. Open the Chrome Developer Tools, specifically the Javascript Console. (see How to Access the Chrome’s Developer Tools section on how to enable this)
9. Examine the output from navigating through the app.

jQuery Mobile Events

Beyond jQuery's $(document).ready() function, jQuery Mobile adds several more events to deal with the framework. They can be grouped as follows:

Touch Events	Simplify working with common touch gestures, like taps, swipes and tap and hold events
Scroll Events	Provides a way to bind to events that fire when scrolling starts and stops
Page Events	Several events for handling page-related issues, such as creation, loading, showing and hiding, and unloading
Orientation Events	Events for handling when the orientation of the device changes.

Page Initialization Events

pagebeforecreate	Triggered when the page is created. Manipulate any markup here before jQM has a chance to do so.
pagecreate	Triggered when the page is created.
pageinit	Triggered when the page is initialized, after jQM has finished enhancing the page content. Use this instead of the usual DOM ready event

Page Transition Events

pagebeforeshow	Triggered on the page that is about to be transitioned to, before the animation is started
pagebeforehide	Triggered on the page that is about to be transitioned away from, before the animation is started
pageshow	Triggered on the page that is being transitioned to, after the animation has completed
pagehide	Trigger on the page that is being transitioned fro, after the animation has completed

Touch Events

tap	Triggered when a user taps on an element
taphold	Triggered when the user taps on an element and holds for about a second
swipe	Triggered when the user swipes either vertically or horizontally
swipeleft	Triggered when the user swipes from right to left
swiperight	Triggered when the user swipes from left to right

PhoneGap Events

Although we are not using these events in this workshop, I thought it would be worthwhile to list the primary events that are PhoneGap specific.

deviceready	The PhoneGap deviceready event fires once PhoneGap has fully loaded. After the device has fired, you can safely make calls to PhoneGap function.
pause	This is an event that fires when a PhoneGap application is put into the background.
resume	This is an event that fires when a PhoneGap application is retrieved from the background.
online	This is an event that fires when a PhoneGap application is online (connected to the Internet).
offline	This is an event that fires when a PhoneGap application is offline (not connected to the Internet).
backbutton	This is an event that fires when the user presses the back button on Android.
menubutton	This is an event that fires when the user presses the menu button on Android.
searchbutton	This is an event that fires when the user presses the search button on Android.

How to Access the Chrome's Developer Tools

To access the developer tools, open a web page or web app in Google Chrome. Then take one of the following actions:

Select the Wrench menu at the top-right of your browser window, then select Tools -> Developer tools.

Right-click on any page element and select Inspect element.

On Windows and Linux, press

Control - Shift - I keys to open Developer Tools

Control - Shift - J to open Developer Tools and bring focus to the Console.

Control - Shift - C to toggle Inspect Element mode.

On Mac, press

⌥⌘I (Command - Option - I) keys to open Developer Tools

⌥⌘J (Command - Option - J) to open Developer Tools and bring focus to the Console.

^⌘C (Control - Option - C) to toggle Inspect Element mode.

Lab 8: Templates

Goal: To leverage templates to reduce page size and increase maintainability

Open the file for this lab.

1. Go to your HTML editor
2. Open File Menu and select Open…
3. Open the file *Lab 8/start/index.html*
4. Add the mustache.js library within the `<head>` section:

   ```
   <script src="../../js/mustache.js"></script>
   ```

5. Now add our JSON data file as well

   ```
   <script src="../../js/npsdata.json"></script>
   ```

6. Now add this script tag

   ```
   <script src="../../js/templates.js"></script>
   ```

7. Save the file, and preview it in the browser.
8. Open the Javascript console and make sure there are no Javascript errors.
9. Add this script tag:

   ```
   <script src="../../js/app.js"></script>
   ```

10. Save the file, and preview it in the browser.

Templating: A Closer Look

Here is the template that is use to generate the list of parks on the first screen of the app:

```
<ul id="parkList" data-role="listview" data-
autodividers="true" data-filter="true">{{#parks}}<li
id="{{id}}"><a href="#parkDetailsPage"><img
src="../../img/thumbs/{{image}}"><h2>{{name}}</h2><p>{{
state}}</p></a></li>{{/parks}}</ul>
```

Mustache uses double curly braces **{{ }}** to denote where the template should inject the data.

`{{#park}}` tells the template to loop over the park array in the nps JSON data. Each item with the array is then injected into the template.

Here is a snippet of the generated HTML (sans the jQueryMobile additions)

```
<ul id="parkList" data-role="listview" data-autodividers="true"
data-filter="true">
<li id="0"><a href="#parkDetailsPage"><img
src="../../img/thumbs/acadia.jpg" c><h2>Acadia</h2><p
>Maine</p></a></li>
<li id="1"><a href="#parkDetailsPage"><img
src="../../img/thumbs/americansamoa.jpg" c><h2>American
Samoa</h2><p >American Samoa</p></a></li>
<li id="2"><a href="#parkDetailsPage"><img
src="../../img/thumbs/arches.jpg" c><h2>Arches</h2><p
>Utah</p></a></li>
<li id="3"><a href="#parkDetailsPage"><img
src="../../img/thumbs/badlands.jpg" c><h2>Badlands</h2><p >South
Dakota</p></a></li>
```

For a great tutorial on Mustache, visit http://coenraets.org/blog/2011/12/tutorial-html-templates-with-mustache-js/

Lab 9: Theming

Goal: to create a custom visual theme

Adding a custom Theme to jQuery Mobile

1. Go to your HTML editor
2. Open File Menu and select Open…
3. Open the file *Lab 9/start/index.html*
4. Replace
5.

```
<link rel="stylesheet" href="../../css/jquery.mobile.css">
```

with

```
<link rel="stylesheet"
href="../../css/jquery.mobile.structure.css">
```

6. After that `<link>` tag, add the following:

```
<link rel="stylesheet" href="../../css/nps.css">
```

Tweaking the Theme

1. After that <link> tag, add the following:

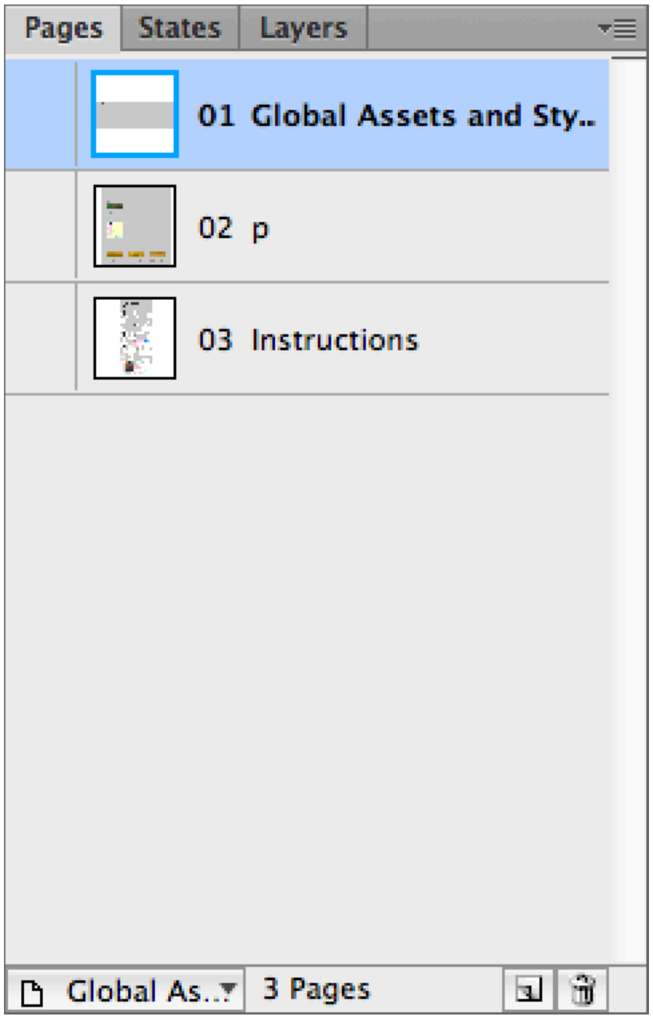
```
<link rel="stylesheet" href="../../css/style.css">
```

2. Open this css file and review the contents.
3. Save the file, and preview it in the browser.

Creating a jQuery Theme in Fireworks

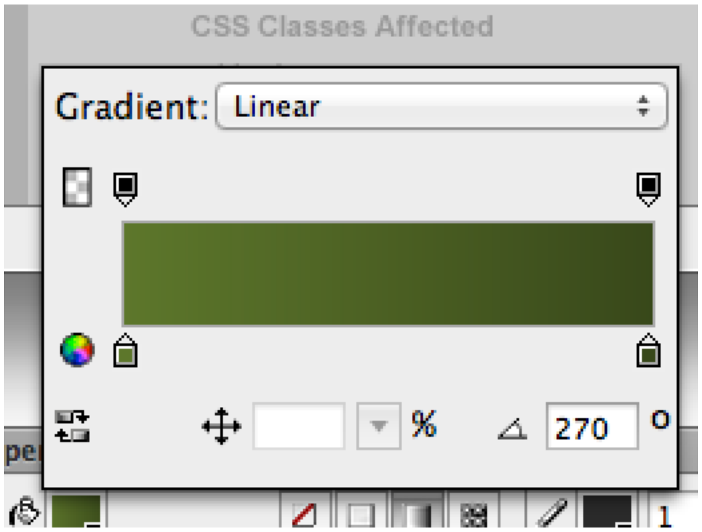
1. Launch **Adobe Fireworks CS6**
2. From the start screen, select *Create New > Fireworks Document (PNG)*
3. Click OK to accept the document size values
4. Go to *Commands > jQuery Mobile Theme > Create New Theme*
5. Select Global Assets and Styles in the Pages Panel

6. Change the Zoom level of the document to 100%
7. Using the Pages Panel, delete pages 03 thru 07 (Theme B-E & Instructions)
8. Select page '**a**', and rename to '**p**'

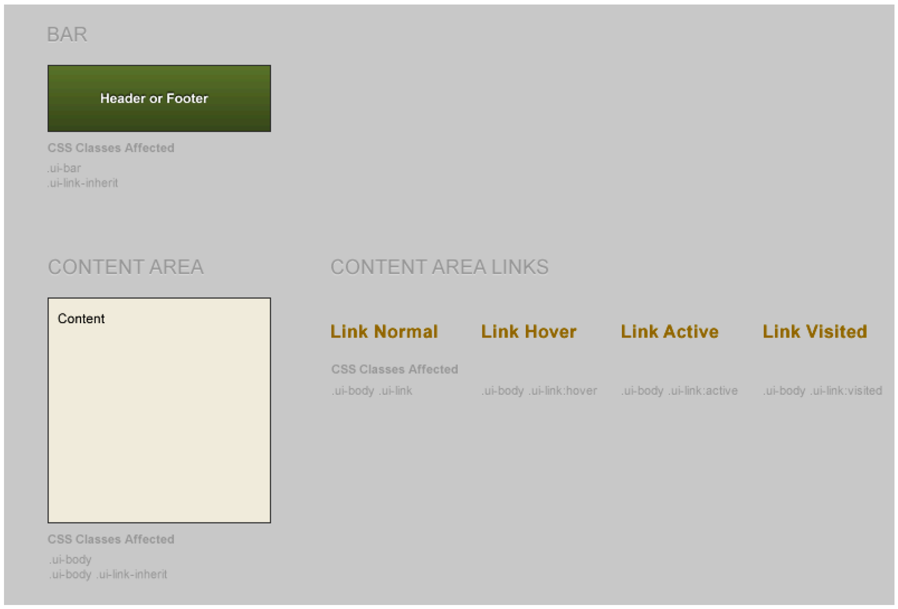


9. Now, select the Bar graphic (Header or Footer)

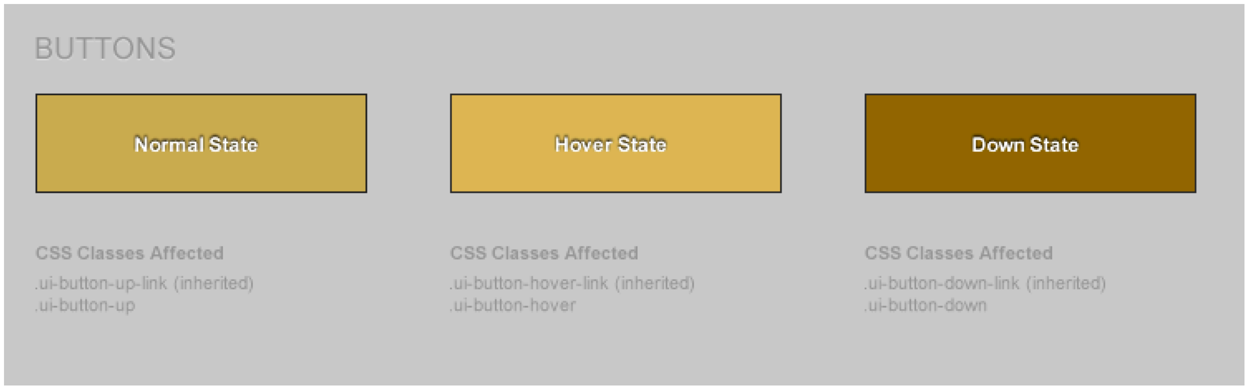
10. In the Properties Panel, click the color chip. Change the starting color to #5e7630 and the ending color to #38471D



11. Now, select the Content Area
12. In the Properties Panel, click the color chip. Change the starting color to #F4EFDD and the ending color to # F4EFDD
13. Select the four Link states and change their color to #916800.



14. Select the Normal State button, and change it's color to start at #CCAC58 and end at #CCAC58.
15. Select the Hover State button, and change it's color to start at #DCB85D and end at #DCB85D
16. Select the Down State button, and change it's color to start at #916800 and end at #916800



17. Save the file *as nps-style.png*
18. Go to *Commands > jQuery Mobile Theme > Preview Theme in Browser*
19. Close this browser window, and return to Adobe Fireworks
20. Go to *Commands > JQuery Mobile Theme > Export Theme*. Save this file in the desktop as *nps.css.*
21. Copy this file into the css directory.

Lab 10: PhoneGap Build

Open the file for this lab.

1. Go to your HTML editor
2. Open File Menu and select Open…
3. Open the file *Lab 10/app/config.xml*
4. Edit the `<author>` node to use your name, email and web url
5. Save the file.
6. Select the app directory, and right-click on it, then *Send To / Compressed (zipped) folder*
7. Open your browser, and go to **build.phonegap.com**
8. Sign in with your AdobeID
9. Click the + New App button
10. Click the Upload a .zip file option.
11. Once, the file has been uploaded and parsed, click the Ready To Build button
12. After a few moments, you apps will be ready for download.

Publishing to Devices

Configuring Your Android Device

1. On your device, select the submenu button and select Settings.
2. Select Applications.
3. Select "Unknown Sources" so non-market apps can be installed.
4. Select Development.
5. Select "USB Debugging".

Some devices may have an extra step or two. Samsung devices, for example, need to have you manually mount them in order for it to connect.

Where to go from here

Build a tablet friendly version
Add Geolocation functions: Live maps or distance to park from current location
Add Camera function: The ability to upload user photos.
Handle offline functionality: Alternate solution for the park map

Completed App

Phone Version

https://build.phonegap.com/apps/329616/install

Tablet Version

https://build.phonegap.com/apps/352838/install

Resources

Sites:
http://jquerymobile.com/
https://build.phonegap.com/docs
http://mustache.js
http://jsonlint.com/
http://www.jslint.com/

Codiqa
An online visual design tool for jQuery Mobile.
http://www.codiqa.com/

ConfiGAP
ConfiGAP is an application designed to help you create the config.xml files used by PhoneGap Build.
http://aj-software.com/configap/

jQuery Mobile Web Development Essentials
by Raymond Camden and Andy Matthews (ISBN: 1849517266)

PhoneGap API Explorer
PhoneGap API Explorer is an API reference application for PhoneGap device integration capabilities. The application clearly presents the syntax for each function, and allows you to provide values for the function's arguments, invoke the function from within the application, and see immediate results. PhoneGap API Explorer provides a unique and interactive learning experience.

Android
https://play.google.com/store/apps/details?id=org.coenraets.phonegapexplorer

iOS
http://itunes.apple.com/us/app/phonegap-api-explorer/id537825489?mt=8

Additional PhoneGap Build features

Hydration

Hydration is a tool that has two main benefits for developers and testers:

- Compilation times are improved significantly.
- Updates are pushed directly to the application installed on a device.

PhoneGap accomplishes this by compiling a native binary that acts as a container for your mobile application. Once a developer uploads a new build the end user (eg. tester) of the application will be notified upon restart of the application. If the end user decides to run the new code base the Hydrated app will automatically fetch and run the latest code base.

https://build.phonegap.com/docs/hydration

Debugging

This service enables users to debug and interactively modify their applications during runtime; this new addition offers similar functionality to those found in Firebug and Google Chrome Inspector, which serve as an indispensable tool to developers working on web based projects.

https://build.phonegap.com/docs/phonegap-debug

Ripple Emulator

The PhoneGap Emulator allows you to test your PhoneGap application from your desktop browser. PhoneGap's JavaScript APIs are available using Ripple, so you can subscribe to deviceready and even stub responses for your custom plugins.

http://emulate.phonegap.com/