Comp 422 - Software Development for Wireless and Mobile Devices

Fall Semester 2016 - Week 3

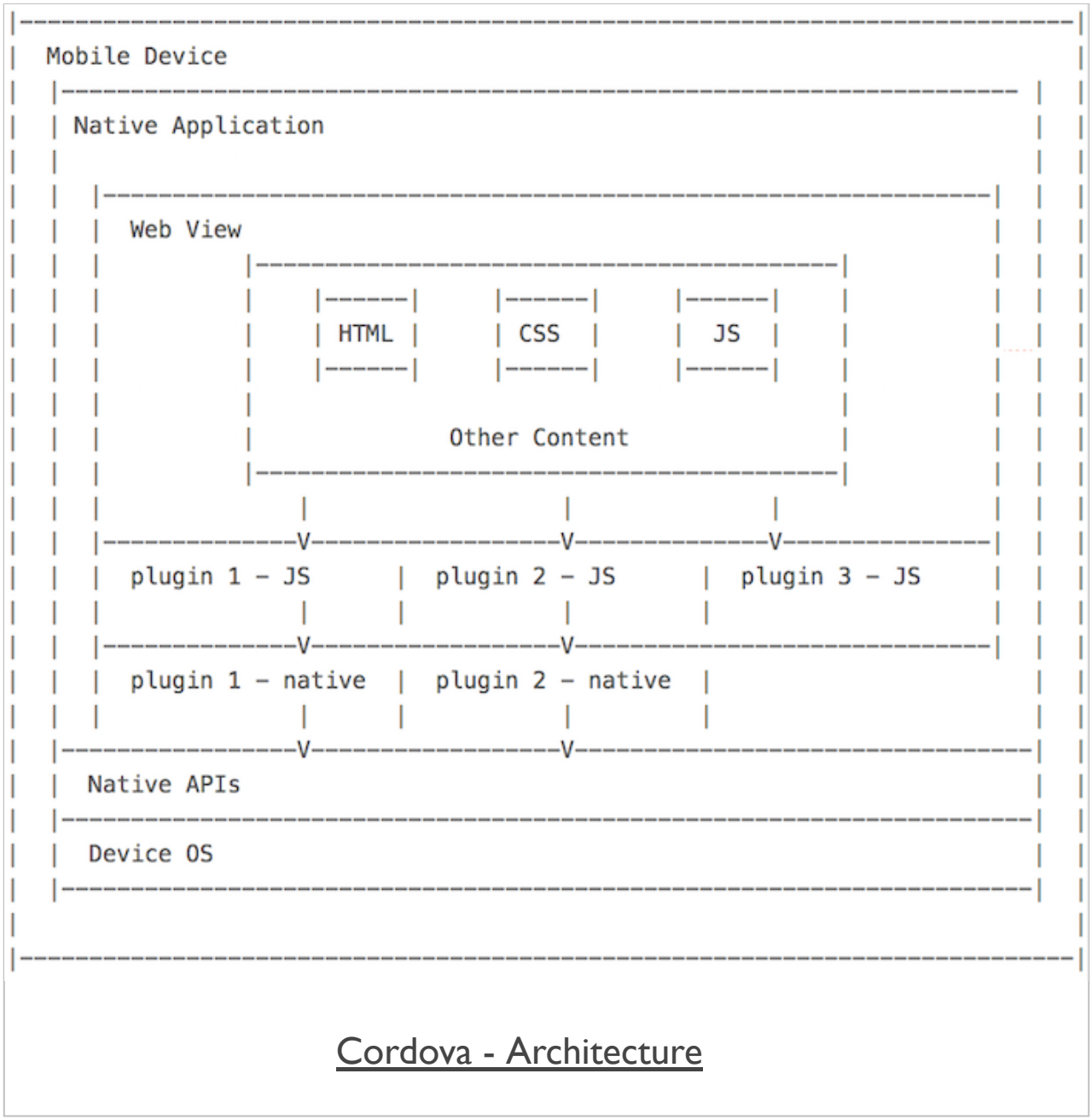
Dr Nick Hayward

Contents

Cordova Design - architecture - intro

- quickly recap the architecture and design behind a Cordova Native application
- Cordova effectively consists of the following components
  - *source code to allow us to build a native application container*
  - *specific to the mobile platforms we choose to add to our project, eg: Android, iOS...*
  - *a collection of various APIs, implemented by Cordova as plugins*
  - *web application running within the container*
  - *access to native device functionality, APIs, and applications*
  - *provides a useful set of tools that help us manage our projects*
  - *creating a project, project files...*
  - *manage required plugins*
  - *build native applications using the native SDK*
  - *testing of applications using emulators, simulators...*

Cordova Design - architecture - diagram



Cordova - Architecture

Cordova Design - architecture

JS & Web plugins

- outline architecture includes the option for JavaScript only plugins
- JS plugins in Cordova normally a bridge from our web container to the native APIs
  - *useful way to expose native device functionality to the web application*
- use and develop plugins purely in JS
  - *add an existing library to help with data visualisations, graphics...*
- create our own focused plugins
  - *abstraction of application features and logic, other specific requirements...*
- greater support for native functionality at the web application level
- HTML5 APIs

Cordova Design - architecture - web container - part 1

- Cordova development uses many of the same underlying technologies as standard web application development
  - *a few limitations relative to network access that we need to consider*
- hybrid mobile application with Cordova
  - *a web application needs to be written as a self-contained application*
  - *needs to be able to run within web container on native device*
  - *constantly fetching external resources not good practice*
  - *mix of local and remote resources preferable for most apps*
  - *external resources an issue if we lose a network connection*
- `index.html` file will normally be the only HTML file we use
  - *separate pages will be containers within this file*

Cordova Design - architecture - web container - part 2

- rethink our approach to building such mobile web stack applications
  - *help us leverage the inherent capabilities of Cordova*
- self-contained applications need to ensure
  - *any application files and data are initially available*
  - *allows the application to launch and load on the native device*
  - *without initial calls to a remote server*
  - *load the application and render the UI*
- application can then optionally fetch data
  - *remote server, API, search query, stream media...*
- consider stages of design for our app's container

Cordova Design - architecture - SDKs and OSs

- build our Cordova applications
  - *including default Cordova APIs or additional APIs*
  - *each app has to be packaged into a native application*
  - *allows app to run on the host native device*
- each native SDK has its own set of custom or proprietary tools
  - *building and packaging their specific native applications*
- build our Cordova applications for a native device
  - *web content portion of app is added to a project*
  - *applicable to the chosen mobile platforms, such as Android, iOS, and Windows Phone*
  - *project is then built for each required platform*
  - *using Cordova CLI, for example*
  - *uses each of the applicable platform specific set of tools to help build*

Cordova App - CLI recap

build initial project

```
cd /Users/ancientlives/Development/cordova
cordova create basic com.example.basic Basic
cd basic
```

- creates new project ready for development

```
cordova platform add android --save
cordova build
```

- adds support for native SDK, Android
- then builds the project ready for testing and use on native device

```
cordova emulate android
```

- outputs current project app for testing on Android emulator

```
cordova prepare android
```

- copies app code into platform ready for building
  - *then use native IDE for build &c...*

Cordova App - structure recap - app directory

- quick recap of app's structure
- new project includes the following default structure

```
|- config.xml
|- hooks
|- README.md
|- platforms
   |- android
   |- platforms.json
|- plugins
|  |- android.json
|  |- cordova-plugin-whitelist
|  |- fetch.json
|- www
|  |- css
|  |- img
|  |- index.html
|  |- js
```

- initially, our main focus will be the www directory

Cordova App - structure recap - www directory

```
|- www
|  |- css
|     |- index.css
|  |- img
|     |- logo.png
|  |- index.html
|  |- js
|     |- index.js
```

Cordova App - basics of development - part I

index.html

```
<html>
    <head>
        <meta http-equiv="Content-Security-Policy" content="default-src 'self'
        data: gap: https://ssl.gstatic.com 'unsafe-eval'; style-src 'self'
        'unsafe-inline'; media-src *">
        <meta name="format-detection" content="telephone=no">
        <meta name="msapplication-tap-highlight" content="no">
        <meta name="viewport" content="user-scalable=no, initial-scale=1,
        maximum-scale=1, minimum-scale=1, width=device-width">
        <link rel="stylesheet" type="text/css" href="css/index.css">
        <title>Hello World</title>
    </head>
    <body>
        <div class="app">
            <h1>Apache Cordova</h1>
            <div id="deviceready" class="blink">
                <p class="event listening">Connecting to Device</p>
                <p class="event received">Device is Ready</p>
            </div>
        </div>
        <script type="text/javascript" src="cordova.js"></script>
        <script type="text/javascript" src="js/index.js"></script>
    </body>
</html>
```

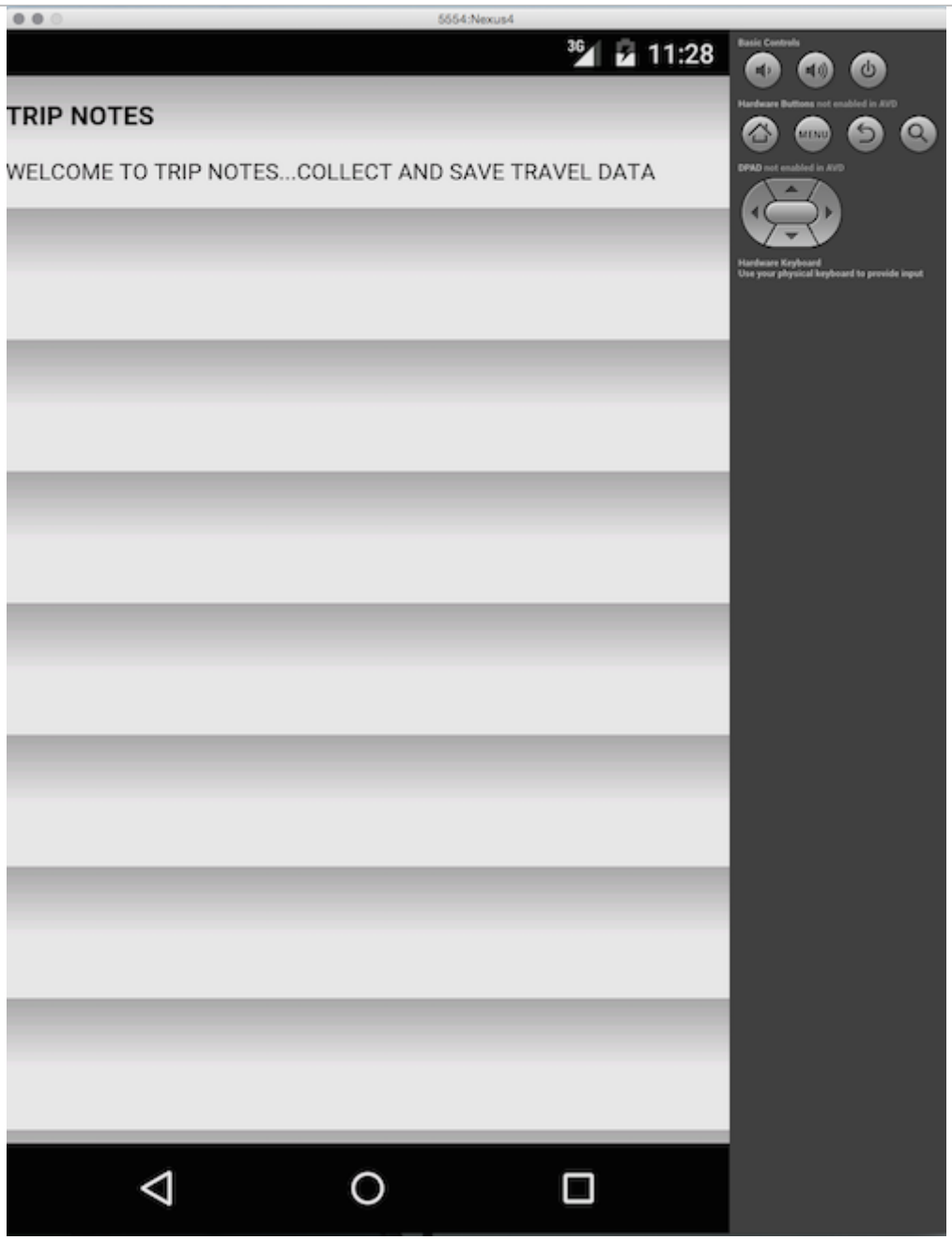
Cordova App - basics of development - part 2

index.html

```
<body>
    <div>
      <h3>Trip Notes</h3>
      <p>
        welcome to trip notes...collect and save travel data
      </p>
    </div>
    <script type="text/javascript" src="cordova.js"></script>
    <script type="text/javascript" src="js/index.js"></script>
</body>
```

- lack of styling will be an issue...

Image - Cordova App - Basic v0.01



Trip Notes - v0.01

Cordova App - basics of development - part 3

add Cordova specifics

- Cordova container for the application exposes native APIs to web application running in WebView
- most APIs not available until applicable plugin added to the project
- container also needs to perform some preparation before the APIs can be used
- Cordova informs us when the container, and associated APIs, are ready for use
- fires a specific event, called the `deviceready` event
- application logic requiring use of Cordova APIs
  - *should be executed after receipt of `deviceready` notification*

Cordova App - basics of development - part 4

add some jQuery

- add to foot of <body>

```
<script type="text/javascript" src="js/jquery.min.js"></script>
```

- add test to `trip.js` file

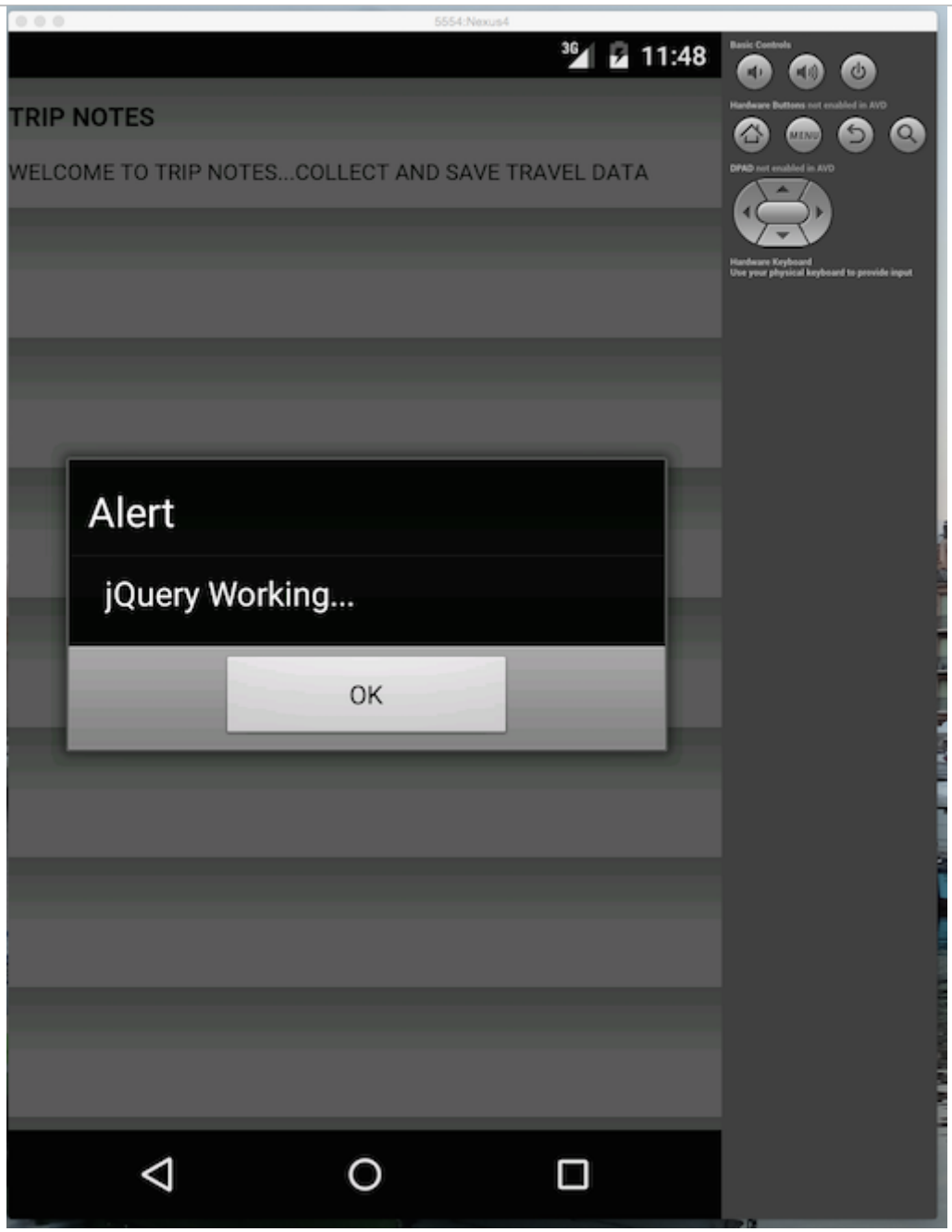
```
function tripNotes() {

  alert("JS Working...");

}


$(document).ready(tripNotes);
```

Image - Cordova App - Basic v0.02



Trip Notes - v0.02

Cordova App - basics of development - part 5

add some jQuery Mobile

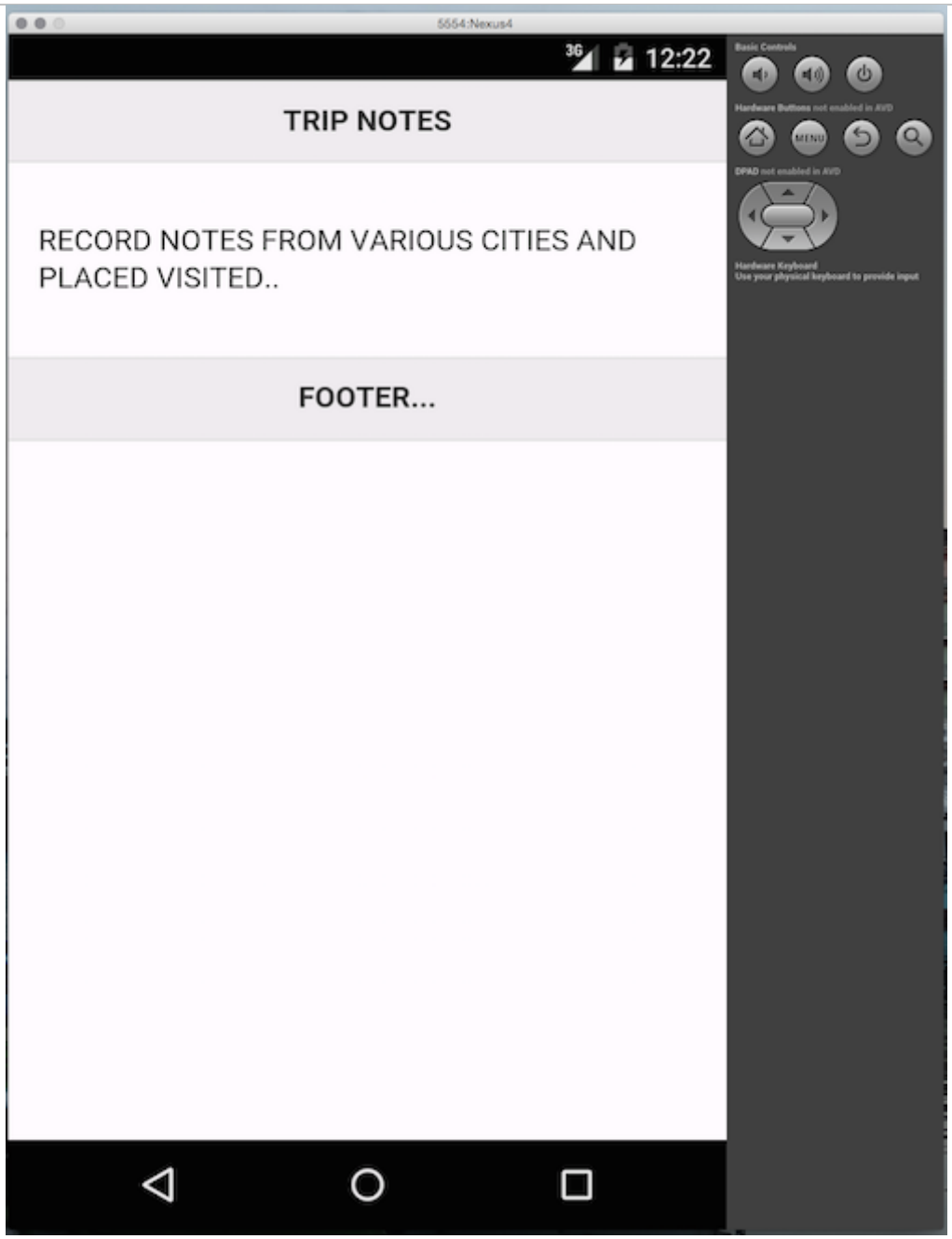
- update head with local jQuery Mobile CSS

```
<head>
...
<link rel="stylesheet" type="text/css" href="css/jquery.mobile.min.css" />
</head>
```

- update body for basic app

```
<body>
  <div data-role="page">
    <div data-role="header">
      <h3>trip notes</h3>
    </div><!-- /header -->
    <div role="main" class="ui-content">
      <p>record notes from various cities and placed visited..</p>
    </div><!-- /content -->
    <div data-role="footer">
      <h5>footer...</h5>
    </div><!-- /footer -->
  </div><!-- /page -->
  <script type="text/javascript" src="cordova.js"></script>
  <script type="text/javascript" src="js/index.js"></script>
  <script type="text/javascript" src="js/jquery.min.js"></script>
  <script type="text/javascript" src="js/jquery.mobile.min.js"></script>
  <script type="text/javascript" src="js/trip.js"></script>
</body>
```

Image - Cordova App - Basic v0.03



Trip Notes - v0.03

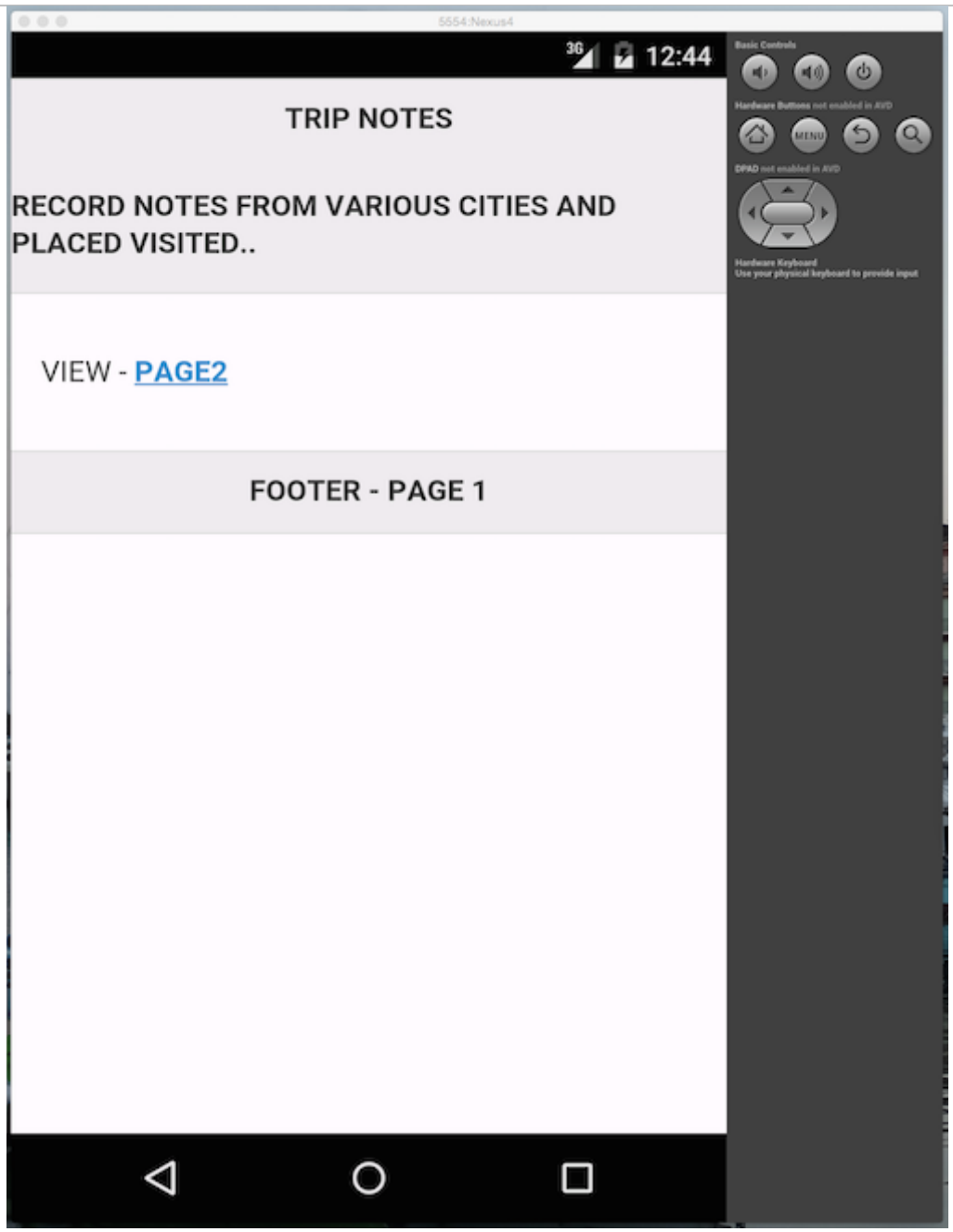
Cordova App - basics of development - part 6

jQuery Mobile - test transitions

- update `index.html` to add page containers, transitions...

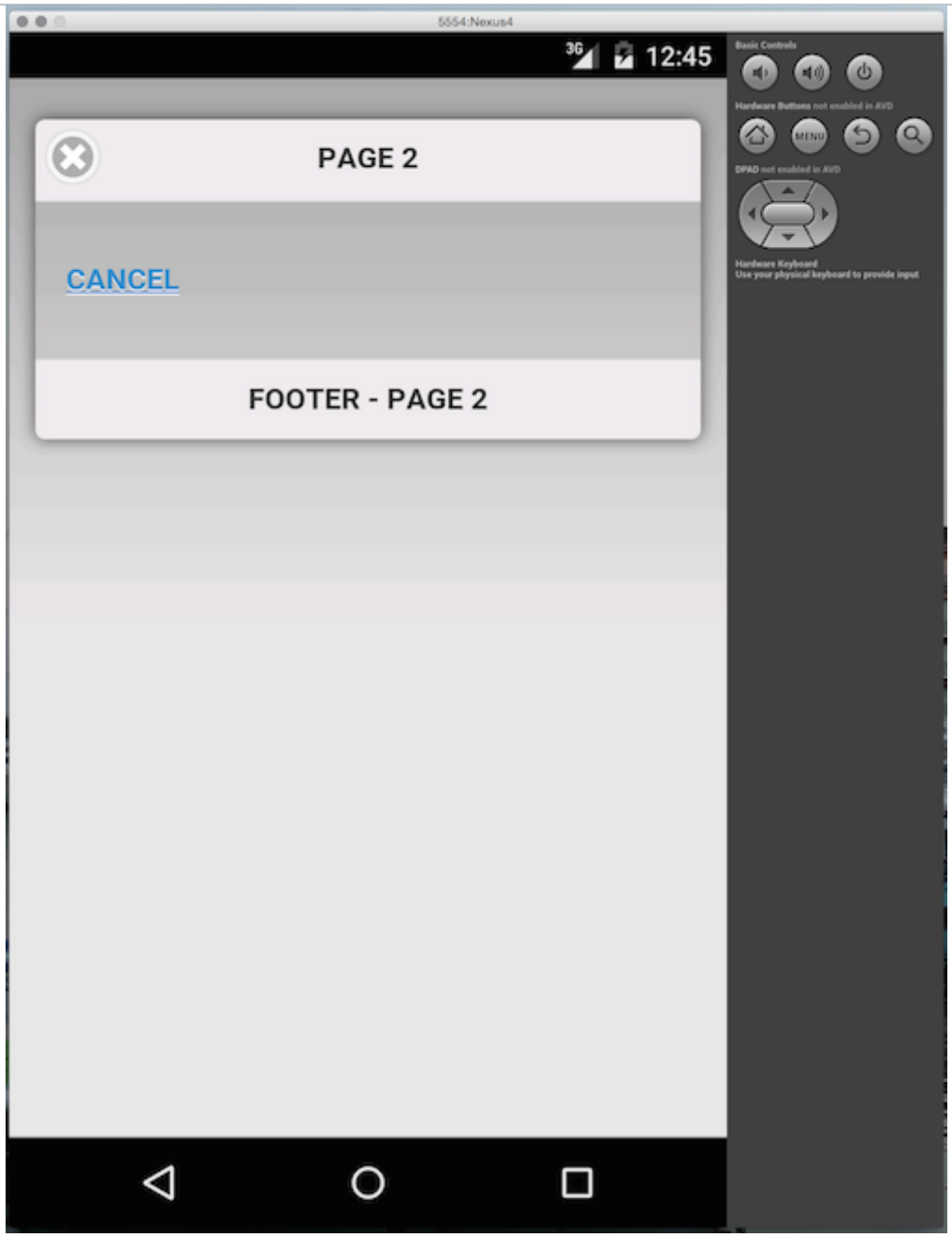
```
<!-- page1 -->
<div data-role="page" id="page1">
  <div data-role="header">
    <h3>trip notes</h3>
    <p>record notes from various cities and placed visited..</p>
  </div><!-- /header -->
  <div role="main" class="ui-content">
    <p>View - <a href="#page2" data-transition="slidedown">page2</a></p>
  </div><!-- /content -->
  <div data-role="footer">
    <h5>footer - page 1</h5>
  </div><!-- /footer -->
</div><!-- /page1 -->
<!-- page2 -->
<div data-role="page" data-dialog="true" id="page2">
  <div data-role="header">
    <h3>page 2</h3>
  </div><!-- /header -->
  <div role="main" class="ui-content">
    <p><a href="#page1" data-rel="back">Cancel</a></p>
  </div><!-- /content -->
  <div data-role="footer">
    <h5>footer - page 2</h5>
  </div><!-- /footer -->
</div><!-- /page2 -->
```

Image - Cordova App - Basic v0.04



Trip Notes - v0.04

Image - Cordova App - Basic v0.05



Trip Notes - v0.05

jQuery Mobile - navigation - part 1

intro

- navigation within our apps
- navigation is thankfully **asynchronous**
- jQuery Mobile navigation loads pages into DOM using AJAX
- modify the page's content, then re-render for display to the user
- includes a set of aesthetically pleasing, and useful, animations
  - *helps inform the user of changes in state, and appropriate content updates*
- navigation system effectively hijacks a link within a page's content container
  - *routes it through an AJAX request*
- benefit for developers is simple approach to asynchronous navigation
- still able to support standard concepts such as **anchors**, **back** button...
  - *without breaking coherence and logic of the application*

jQuery Mobile - navigation - part 2

intro - continued

- jQuery Mobile is able to load and view groups of disparate content
  - *using page content containers within our initial home document*
- support for core JavaScript event handling
  - *URL fragment identifiers with `hashchange` and `popstate`*
- allows the application to persist navigation history, at least temporarily
  - *a record of user navigation and paths through the content*
- tap into this internal history of the application
  - *hijack certain patterns to help us better inform the user*
  - *add details about state changes, different paths, content, and so on...*

jQuery Mobile - navigation - part 3

example navigation

- example of using the jQuery Mobile standard method,

```
$.mobile.navigate
```

- used as a convenient way to track history and navigation events
- set our record information for the link
  - *any useful information for the link or affected change in state*
- log the available direction for navigation
- url for the nav state, and any available hash
  - *in our example the simple hash, `#nav1`*
- Demo - jQuery Mobile nav

jQuery Mobile - using widgets - part I

- within our app's webview
  - *add standard HTML elements for content containers*
  - *use HTML, HTML5...*
  - *e.g. `<p>`, `<h1>, <h2>...`, `li`, `<section>`...*
- jQuery Mobile includes a wide-range of widgets
- simply add the widgets to our applications
- touch friendly widgets
  - *eg: collapsible elements, forms, responsive tables, dialogs...*
  - *`pageContainer` widget for a content container*

jQuery Mobile - using widgets - part 2

listviews

- style, render, manipulate standard data output and collections
- render lists as interactive, animated views
- lists are coded with a `data-role` attribute
  - *similar to structure for a page...*

```
data-role="listview"
```

- we can also set links on our lists
  - *rendered with styling and link icons*
  - *add new page, add extra styles...*
- Demo - jQuery Mobile listview 1
- Demo - jQuery Mobile listview 2

jQuery Mobile - using widgets - part 3

listviews - example

```
<!-- listview example -->
<div>
  <ul data-role="listview">
    <li>Cannes</li>
    <li>Marseille</li>
    <li><a href="#page3" data-transition="slide">Monaco</a></li>
    <li>Nice</li>
  </ul>
</div>
```

- simple listview with slide transition

```
<!-- page3 -->
<div data-role="page" id="page3">
  <div data-role="header">
    <h3>page 3</h3>
  </div><!-- /header -->
  <div role="main" class="ui-content">
    <p><a data-rel="back" class="ui-btn">Return</a></p>
  <section class="image-view">
    <img src="media/images/monaco1.jpg">
  <section>
  </div><!-- /content -->
  <div data-role="footer">
    <h5>footer - page 3</h5>
  </div><!-- /footer -->
</div><!-- /page3 -->
```

- new page for Monaco image
- Demo - jQuery Mobile listview 3

jQuery Mobile - using widgets - part 4

listviews

- use listviews to add filtering and live search options to our lists
- set a simple client-side filter
  - *add an attribute for* `data-filter`
  - *then set the value to* `true`

```
data-filter="true"
```

- also set some default, helpful text for the input field
  - *prompts user to interact, and use this feature correctly*

```
data-filter-placeholder="Search Cities"
```

- tidy up the presentation of our list, add an inset using the attribute

```
data-inset="true"
```

- Demo - jQuery Mobile listview 4

jQuery Mobile - using widgets - part 5

listviews - adding some formatted content

- fun aspects of working with a framework such as jQuery Mobile
  - *simple way we can organise, format our data presentations and views*
- grouped dataset can still be presented using lists
  - *add informative headings*
  - *links to different categories within this dataset*
  - *add simple styling to help differentiate list components*
- structure the list as normal, with sub-headings, paragraphs, and so on
  - *jQuery Mobile option for setting list content as an aside*

```
<p class="ui-li-aside">1 image</p>
```

- many similar tweaks, additions for listviews...
  - *visit jQuery Mobile API for further details*

jQuery Mobile - using widgets - part 6

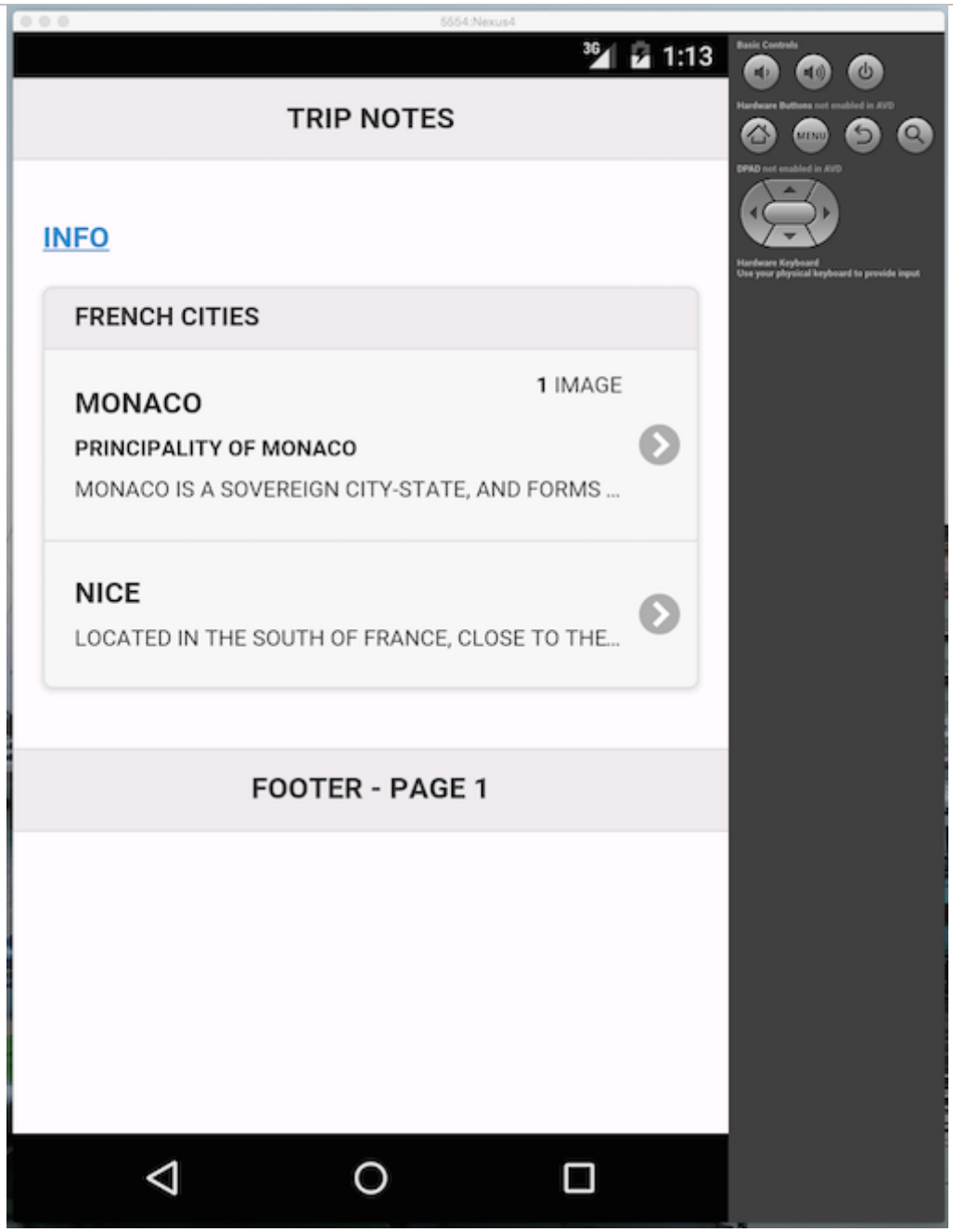
listviews - example

```
<ul data-role="listview" data-inset="true">
  <li data-role="list-divider" role="heading">French Cities</li>
  <li>
    <a href="#page3" data-transition="slide">
      <h3>Monaco</h3>
      <p><strong>Principality of Monaco</strong></p>
      <p>Monaco is a sovereign city-state, and forms part of the French Riviera...</p>
      <p class="ui-li-aside"><strong>1</strong> image</p>
    </a>
  </li>
  <li>
    <a href="#">
      <h3>Nice</h3>
      <p>Located in the south of France, close to the border with Italy...</p>
    </a>
  </li>
</ul>
```

- Demo - jQuery Mobile listview 5

Cordova App - basic - part 7

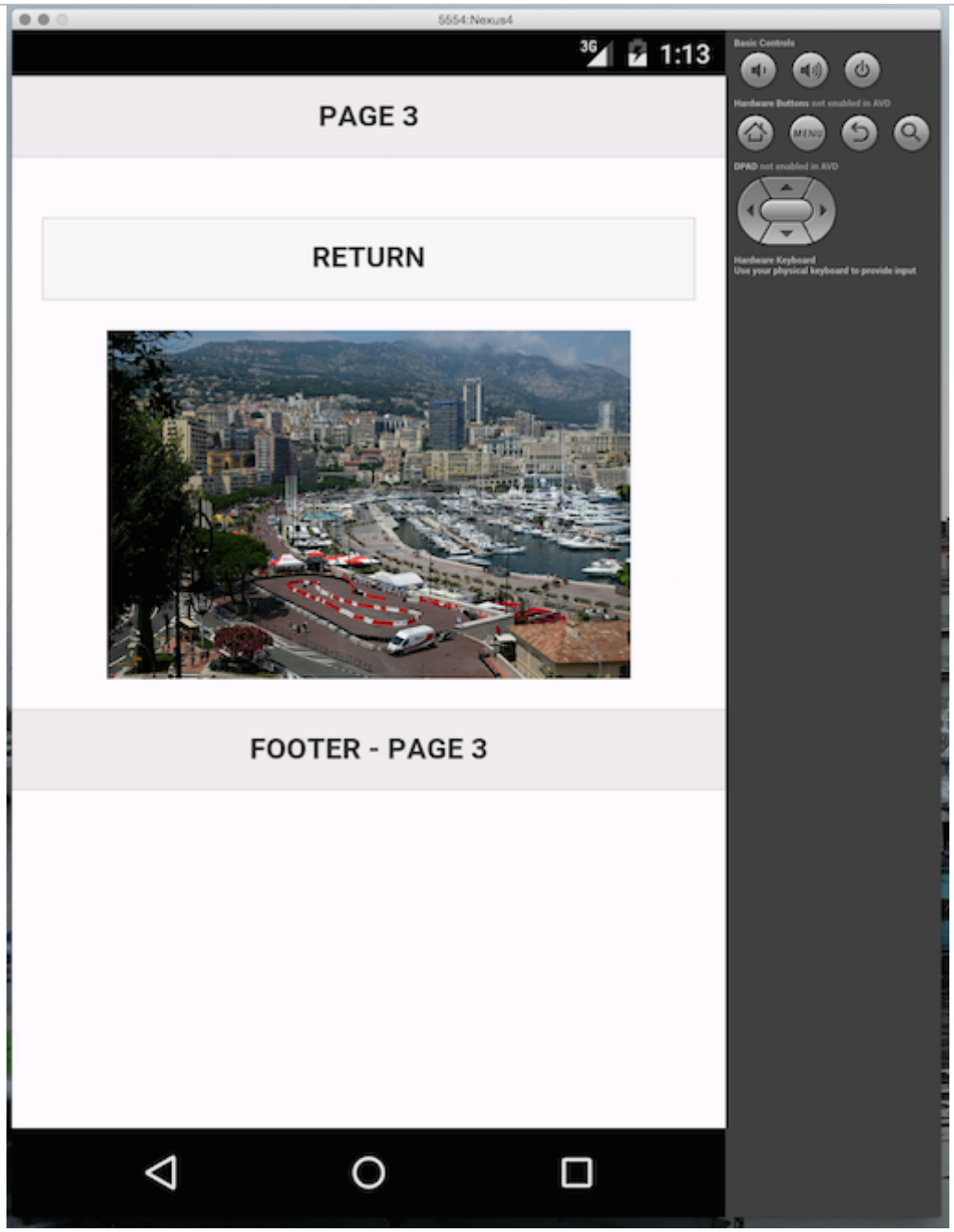
jQuery Mobile - add some organisation



Trip Notes - v0.06

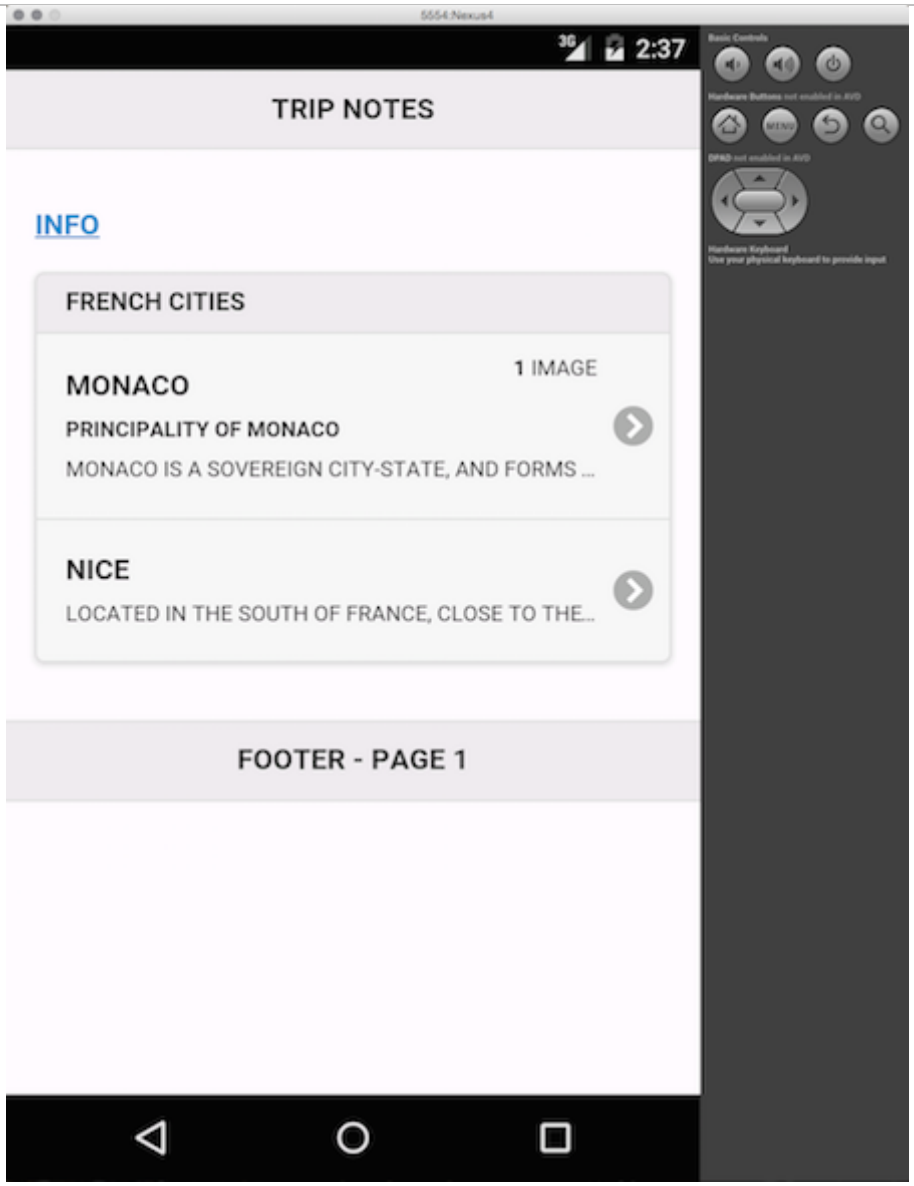
Cordova App - basic - part 8

jQuery Mobile - add some organisation



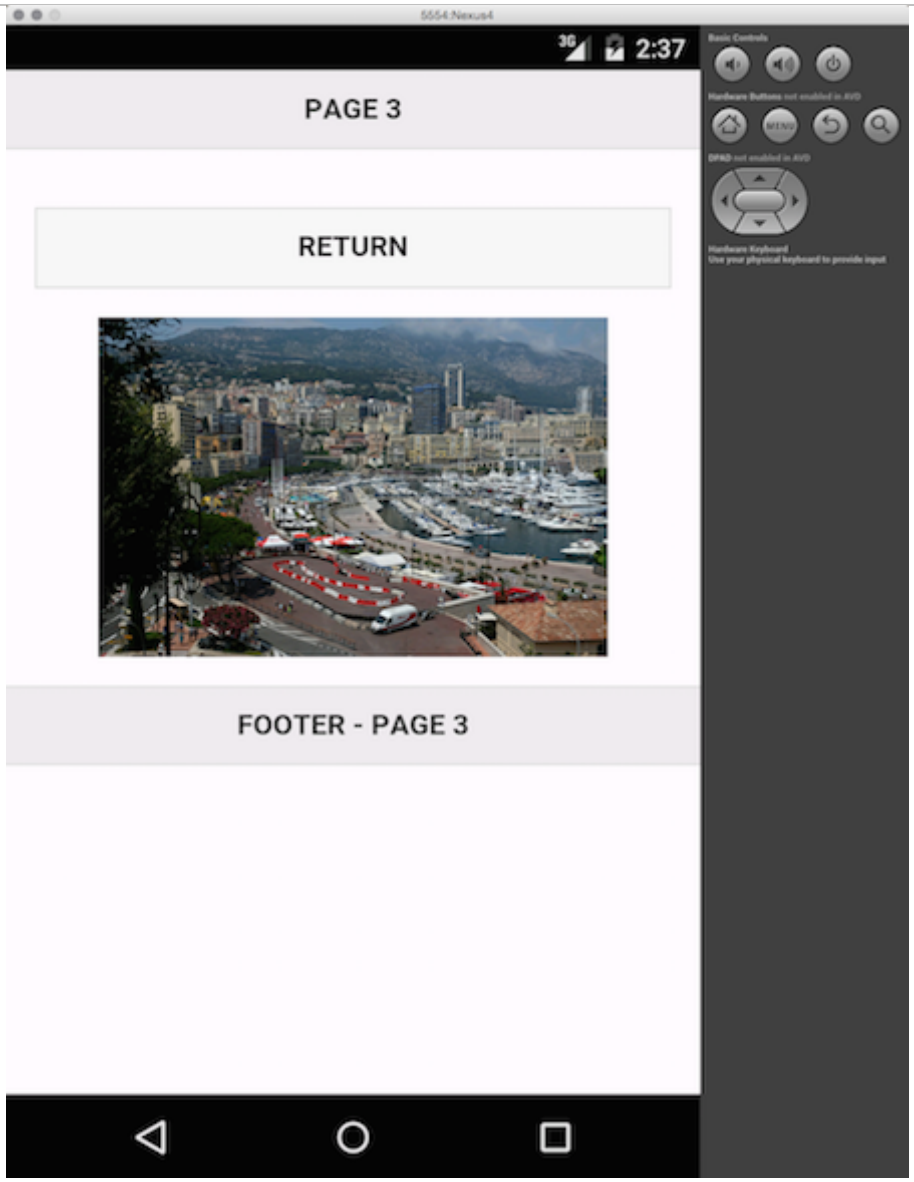
Trip Notes - v0.07

Image - Cordova app - Trip Notes - example 1



Cordova - trip notes - home screen

Image - Cordova app - Trip Notes - example 2



Cordova - trip notes - page with image

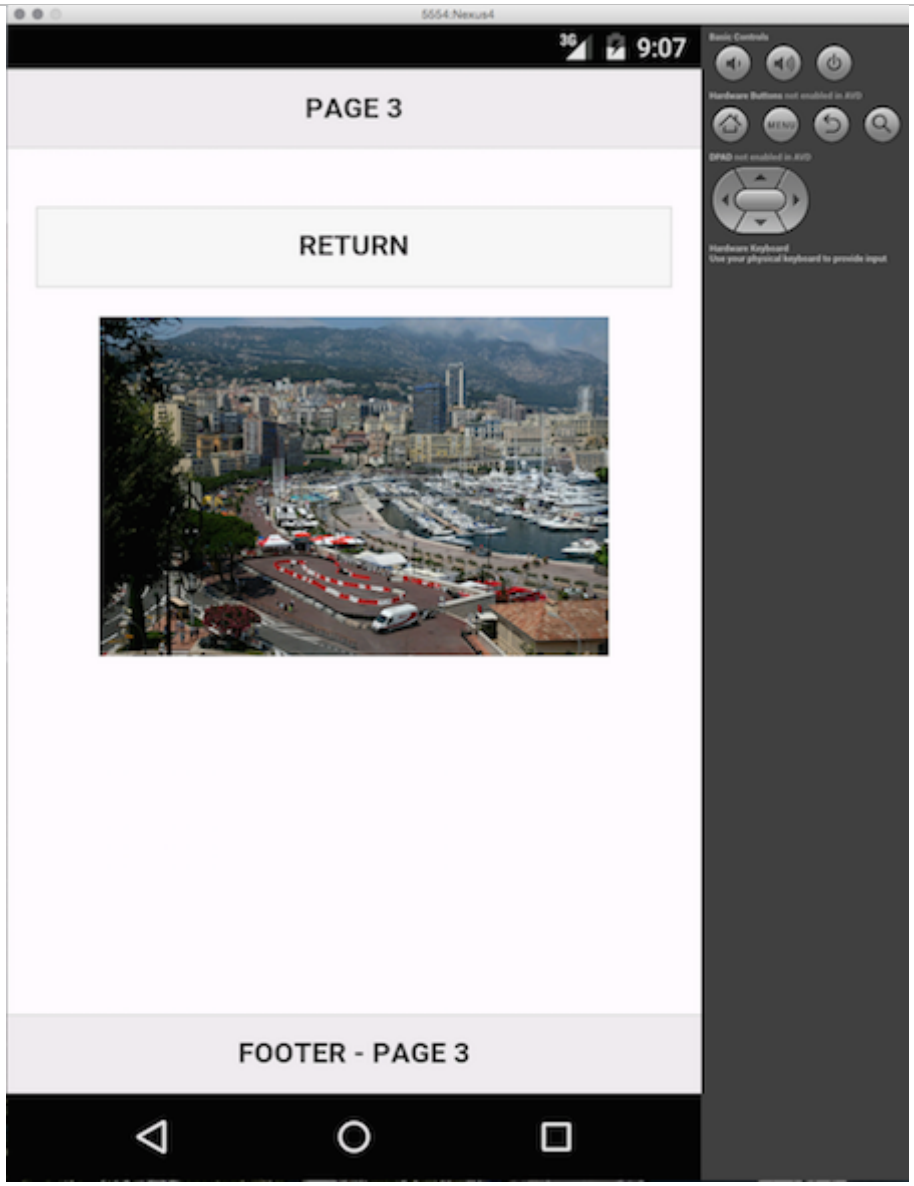
Cordova app - current design

- current design includes
  - *header*
  - *main*
  - *footer*
- fix footer to bottom of a given view using the following attribute,

```
<div data-role="footer" data-position="fixed">
  <h5>footer - page 3</h5>
</div><!-- /footer -->
```

- set this attribute on any of our footer sections

Image - Cordova app - Trip Notes - example 3



Cordova - trip notes - fixed footer

Cordova app - create shell app

blueprint

- create a shell app we can use as a template
  - *use with initial designs & jQuery Mobile*
- helps with initial project development
  - *updating designs*
  - *testing new features*
  - *working with various APIs...*
- updates include
  - *layout of* `index.html`
  - *a few custom styles for* `style.css`
  - *then add an initial splash screen and settings*
  - *then add an initial app icon...*

Cordova app - settings - `config.xml`

blueprint

- an Apache filled `config.xml` file
  - *we need to slightly modify for our requirements*

```
<name>blueprint</name>
<description>
    blueprint for Apache Cordova frameworks with jQuery Mobile
</description>
<author email="test@test.com" href="http://csteach422.github.io">
    ancientlives
</author>
<content src="index.html" />
```

- update <access> element for production usage...

Cordova app - `index.html`

blueprint

- good idea to strip out the `index.html` page
  - *create a consistent layout and structure for developing applications*
- start with a simple body
  - *organised with a header and a main content category*

```
<!-- homepage -->
<div data-role="page" id="home">
    <div data-role="header">
    <h3>blueprint</h3>
    </div><!-- /header -->
  <div role="main" class="ui-content">


    </div><!-- /content -->
</div><!-- /home -->
```

- many mobile applications do not include a footer within their content categories
  - *unless specifically required by a given application structure or functionality*
  - *leave footer out of this default blueprint*
  - *add footer as needed to app's specific template*

Cordova app - `style.css`

blueprint

- for initial applications use default styling offered by jQuery Mobile
  - *otherwise, we can remove all defaults*
  - *create our own default, basic styles*
- preferred aesthetic scheme and palette
  - *add to app's custom CSS file*

Cordova app - working with plugins - getting started

- start looking at some of the plugins available for Cordova
  - *media playback*
- test our initial Cordova blueprint with jQuery Mobile
  - *add some existing plugins*
  - *see how they fit together to create a coherent, basic application*
- create our new project

```
cordova create plugintest1 com.example.plugintest plugintest1
```

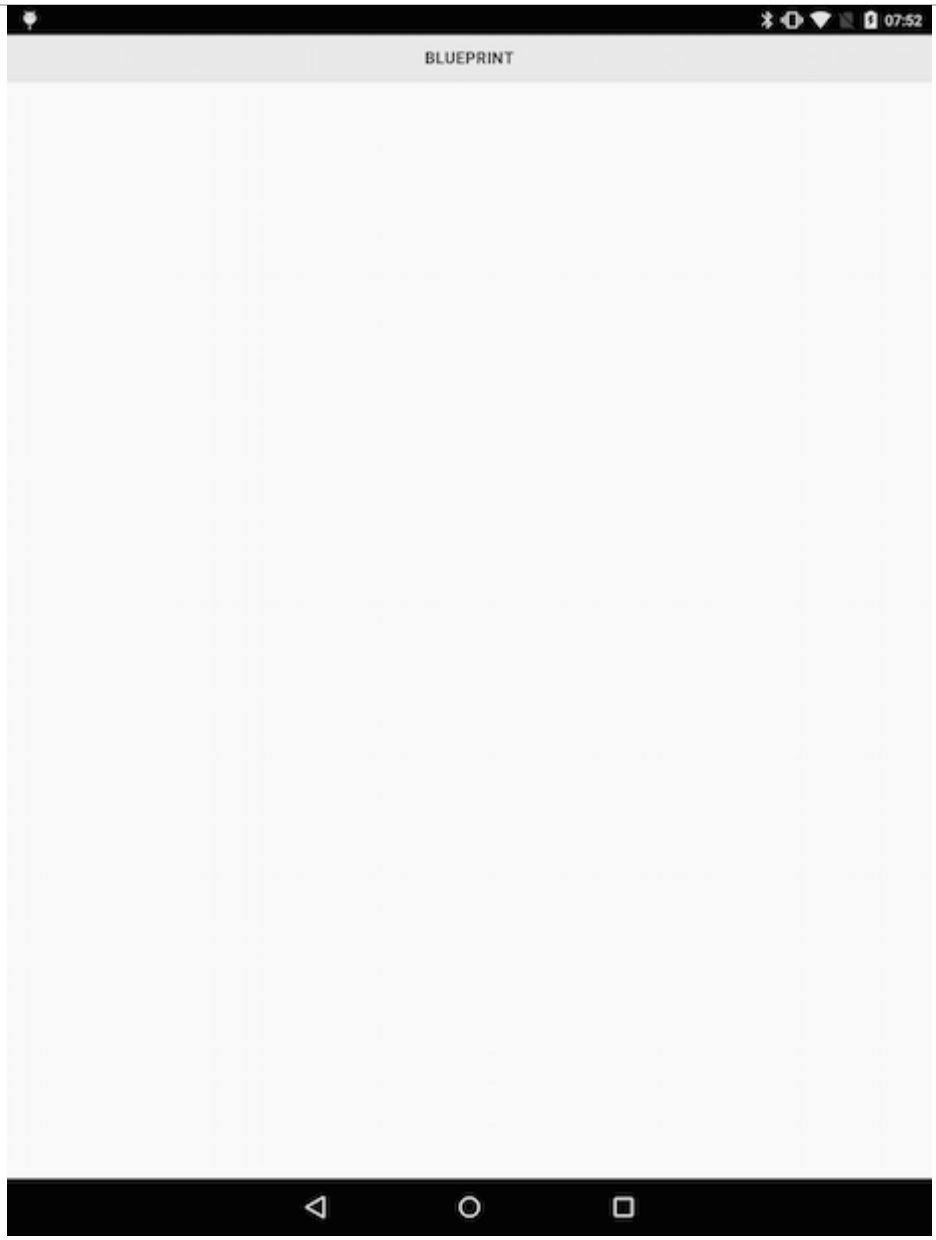
- add support for Android platform

```
cordova platform add android --save
```

- add support for other platforms, as required, such as iOS, Windows...
- transfer our default www directory from the **blueprint**
- start updating some of the settings in the `config.xml` file for the application
  - *metadata for author, description, name...*
- quickly run and test this base for our new application

```
//run in the Android emulator
cordova emulate android
//run on a connected Android device
cordova run android
```

Image - Cordova app - Plugin Test 1 - getting started



Cordova - Plugin Test - getting started

Cordova app - working with plugins - add plugins

- add our required plugins to the test application
  - *add plugins for **device**, **file**, and **media***
- **device** plugin added to check and read information about current device
  - *in effect our Android phone or tablet*
- **file** plugin is required to access the device's underlying filesystem
- **media** helps us record and playback media files
- add these plugins to our project with the following Cordova commands

```
//add device plugin - Git and NPM options
cordova plugin add https://git-wip-us.apache.org/repos/asf/cordova-plugin-device.git
cordova plugin add cordova-plugin-device
//add file plugin - Git and NPM options
cordova plugin add https://git-wip-us.apache.org/repos/asf/cordova-plugin-file.git
cordova plugin add cordova-plugin-file
//add media plugin - Git and NPM options
cordova plugin add https://git-wip-us.apache.org/repos/asf/cordova-plugin-media.git
cordova plugin add cordova-plugin-media
```

- ensure new plugins are applied to our current project
  - *run the following Cordova command*

```
cordova build
```

***n.b.** NPM plugin install is now recommended for latest Cordova apps*

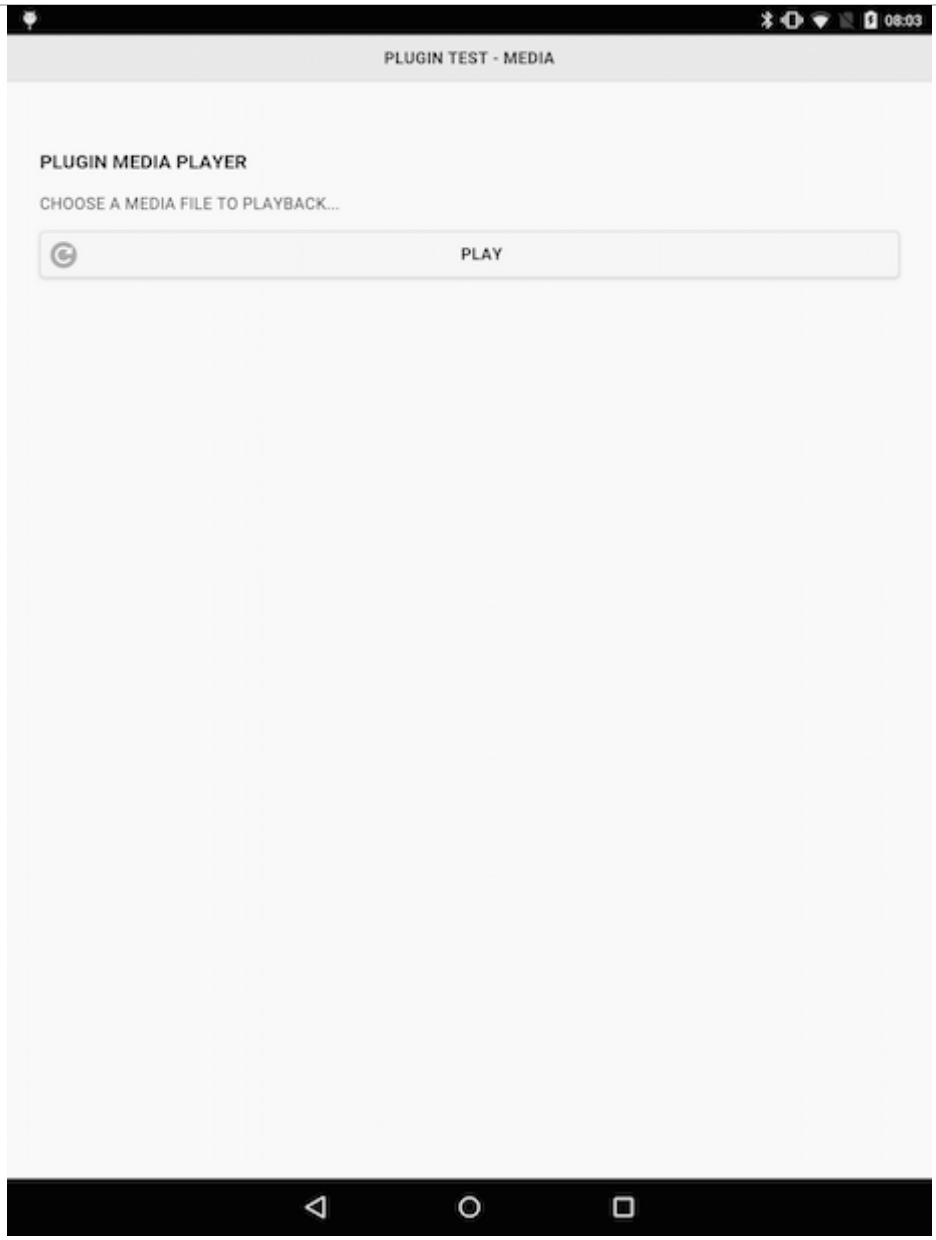
Cordova app - working with plugins - update `index.html`

- update our `index.html` page to create the basic layout
  - *allow us to load and use media files*
- use a single page application structure
  - *include our content categories for `header` and `main`*
- add `div` with `data-role` set to `fieldcontain`
  - *signifies that we have a contiguous group of form, input elements*
- use this grouping to add our **play** button
  - *load our sample file using the installed plugins*
- use an `input` element with type set to button
  - *perhaps add an icon*

Cordova app - working with plugins - `index.html` page structure

```
<!-- homepage -->
<div data-role="page" id="home">
    <div data-role="header">
    <h3>plugin test - media</h3>
    </div><!-- /header -->
  <div role="main" class="ui-content">
    <!-- container for media options... -->
    <div data-role="content">
      <!-- group buttons etc -->
      <div data-role="fieldcontain">
        <h3>Plugin Media Player</h3>
        <p>choose a media file to playback...</p>
        <input type="button" id="playAudio" data-icon="refresh" value="Play" />
      </div>
    </div>
    </div><!-- /content -->
</div><!-- /homepage -->
```

Image - Cordova app - Plugin Test 1 - getting started



Cordova - Plugin Test - index.html

Cordova app - working with plugins - add some logic

- add some logic to our application
- updates to our JavaScript to allow us to handle events
- add handlers for listeners for each button we add to the application
  - *including the initial **play** button*
- add this code to our application's custom JavaScript file
  - `plugin.js`
- setup the application in response to Cordova's `deviceready` event
  - *event informs us that installed plugins are loaded and ready for use*
- add a function for the `deviceready` event
  - *allows us to bind our handler for the tap listener on the **play** button*

```
functon onDeviceReady() {
  $("#playAudio").on("tap", function(e) {
    //add code for action...
  });
}
```

Cordova app - working with plugins - `onDeviceReady()`

- add any other required, initial functions later to this same start-up function
- wrap initial function in our main application loader
  - *checks device is ready, and then adds any required handlers*

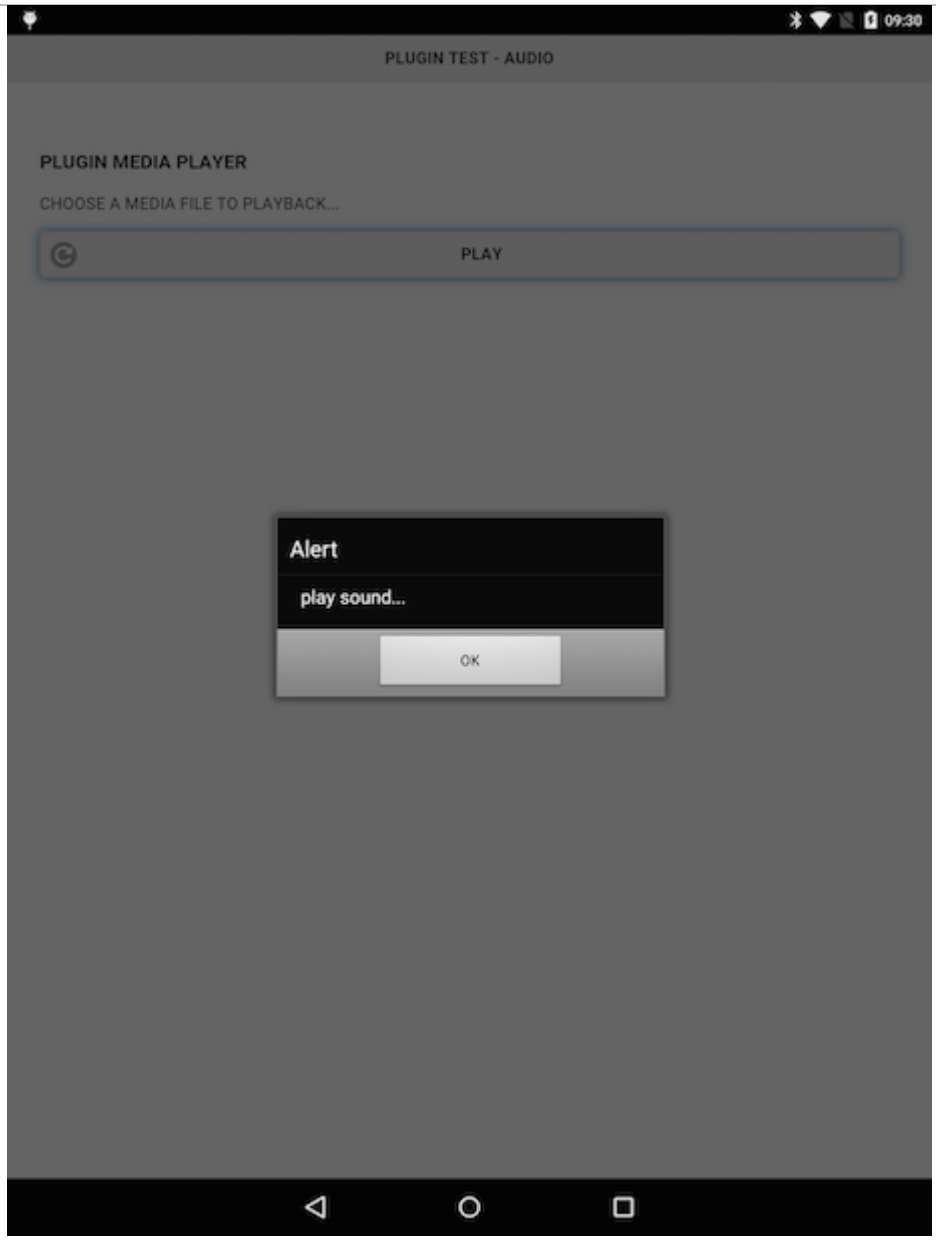
```
(function() {
    //check for page initialisation and #home
    $(document).on("pageinit", "#home", function(e) {
        //prevent any bound defaults
        e.preventDefault();

        //loader function after deviceready event returns
        function onDeviceReady() {
            //play audio
            $("#playAudio").on("tap", function(e) {
                //audio playback logic
                alert("play sound...");
            });

        }
        //as deviceready returns load onDeviceReady()
        $(document).on("deviceready", onDeviceReady);
    });

})();
```

Image - Cordova app - Plugin Test 1 - getting started



Cordova - Plugin Test - audio button

Cordova app - working with plugins - audio playback logic

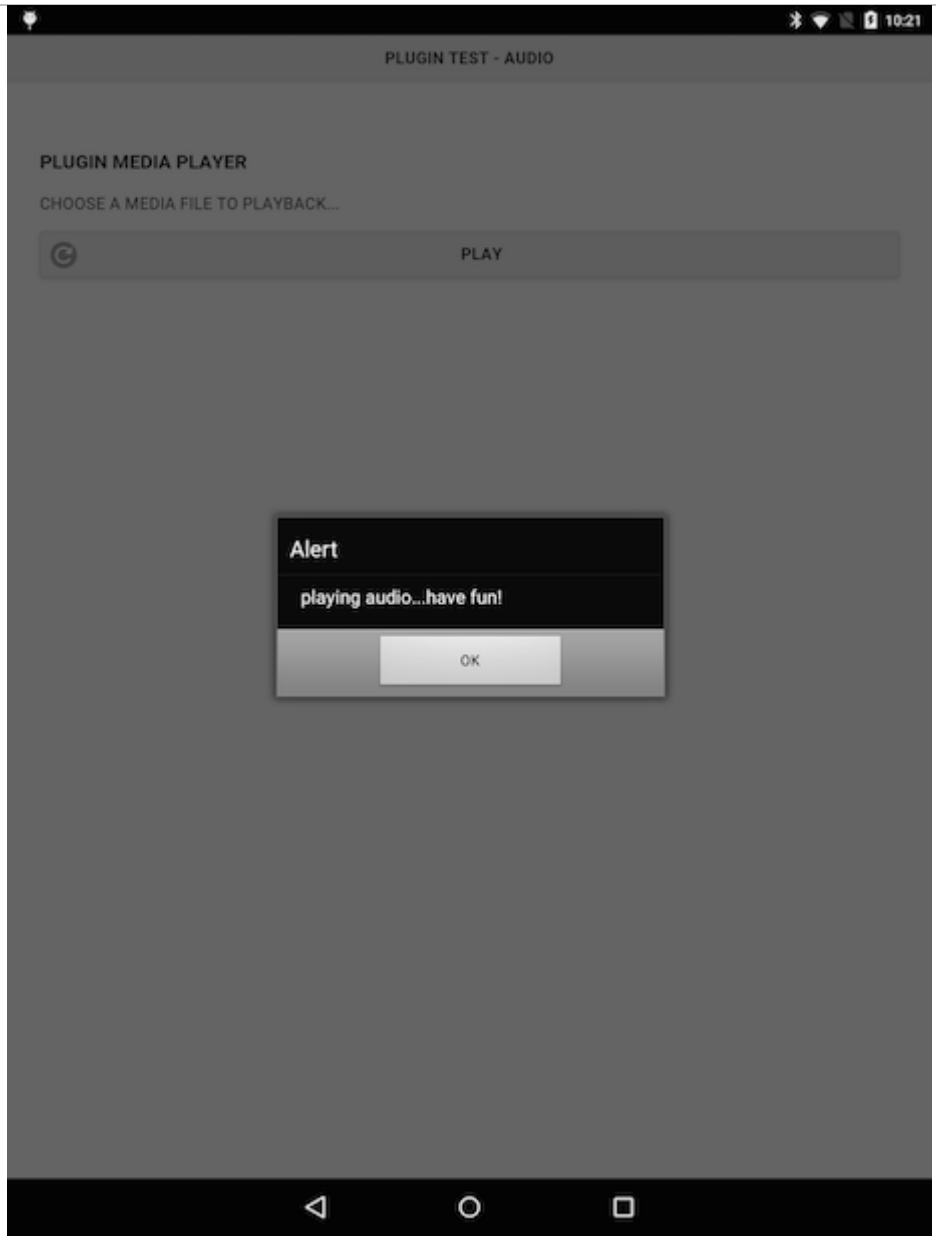
- now setup and tested the basic app logic
  - *added handlers for `deviceready` and clicking the audio playback button*
- update logic for the `#playAudio` button

```
//play audio file
function playAudio() {
  //initial url relative to WWW directory - then built for Android
  var $audioURL = buildURL("media/audio/egypt.mp3");
  var $audio = new Media($audioURL, null, errorReport);
  $audio.play();
  alert("playing audio...have fun!");
}
```

- add associated media loaders for the audio file
- add basic error checks in case the media file is missing, corrupt...

```
//build url for android
function buildURL(file) {
  if (device.platform.toLowerCase() === "android") {
    var $androidFile = "/android_asset/www/" + file;
    return $androidFile;
  }
}
//return any error message from media playback
function errorReport(error) {
  alert("Error with Audio - " + JSON.stringify(error));
}
```

Image - Cordova app - Plugin Test 1 - getting started



Cordova - Plugin Test - audio playback

Demos - jQuery Mobile

- Demo - jQuery Mobile nav
- Demo - jQuery Mobile listview 1
- Demo - jQuery Mobile listview 2
- Demo - jQuery Mobile listview 3
- Demo - jQuery Mobile listview 4
- Demo - jQuery Mobile listview 5

References

- Cordova
  - *Plugin - Device*
  - *Plugin - File*
  - *Plugin - Media*
- jQuery Mobile API
  - *Pagecontainer widget*
- jQuery Mobile 1.4 Browser Support
- jQuery ThemeRoller
- MDN - default event
- Responsive Web Design