Comp 322/422 - Software Development for Wireless and Mobile Devices

Fall Semester 2017 - Week 3

Dr Nick Hayward

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
|- res
|  |- icon
|  |- screen
|- 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
```

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>
```

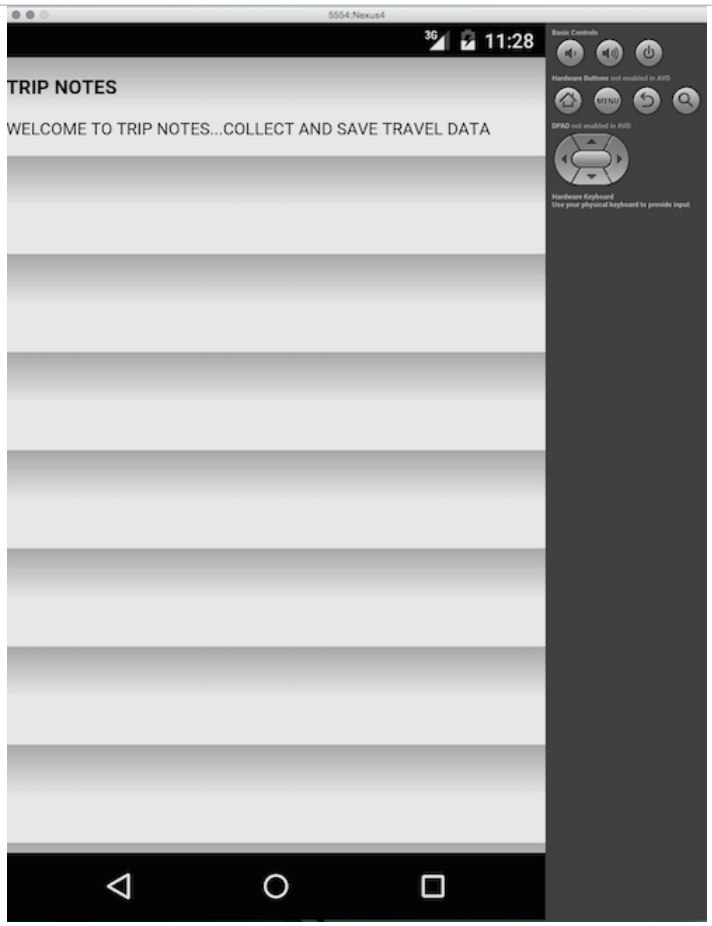
Cordova App - basics of development - part 2

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>

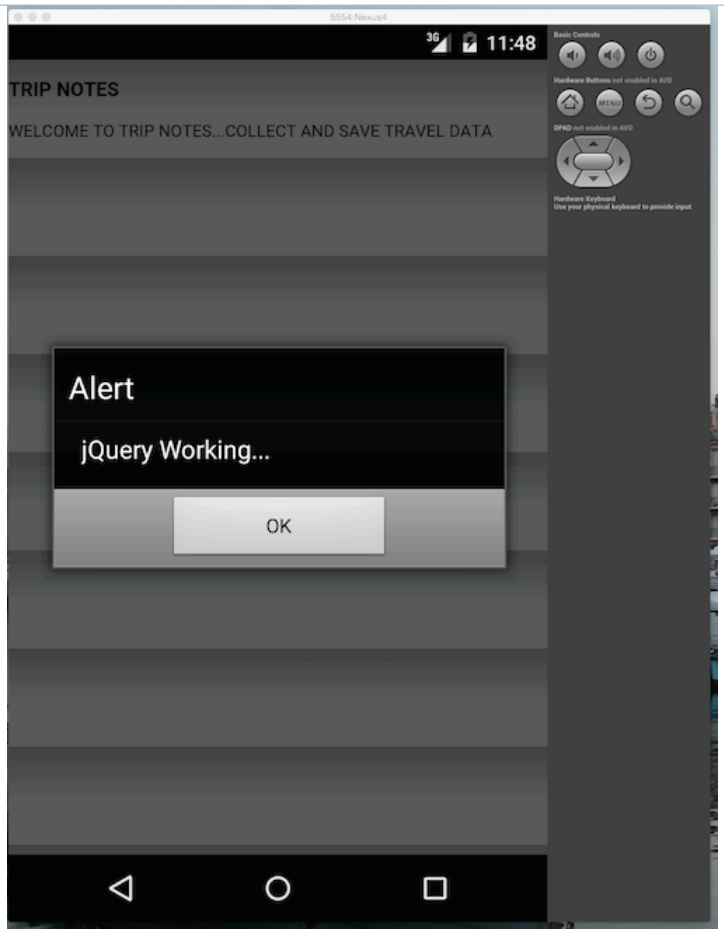
```
<script type="text/javascript" src="js/jquery.min.js"></script>
```

- 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

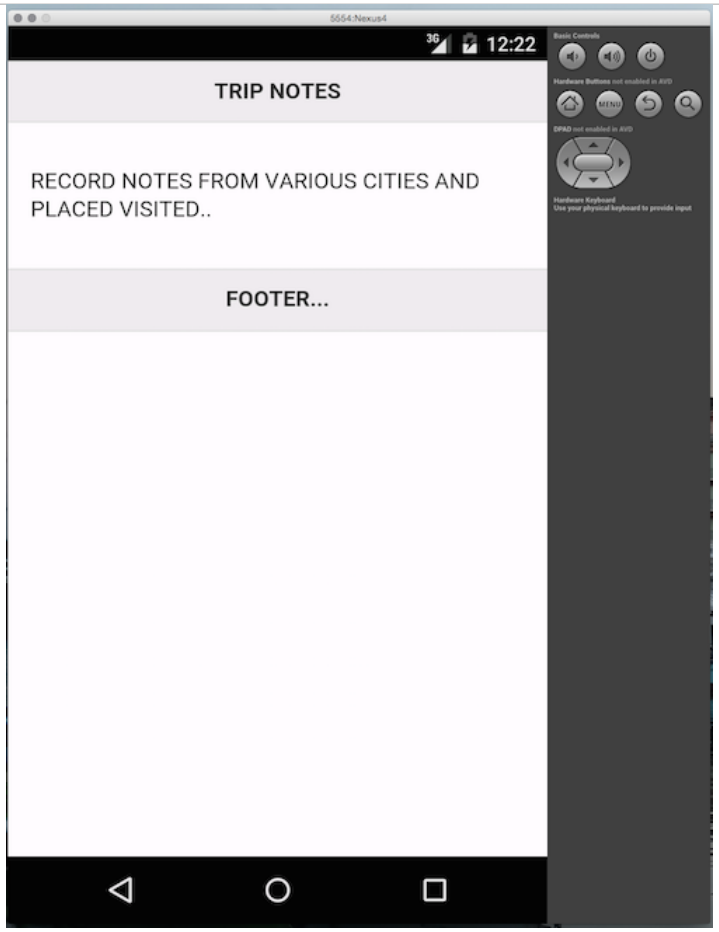
- update head with local jQuery Mobile CSS

```
<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

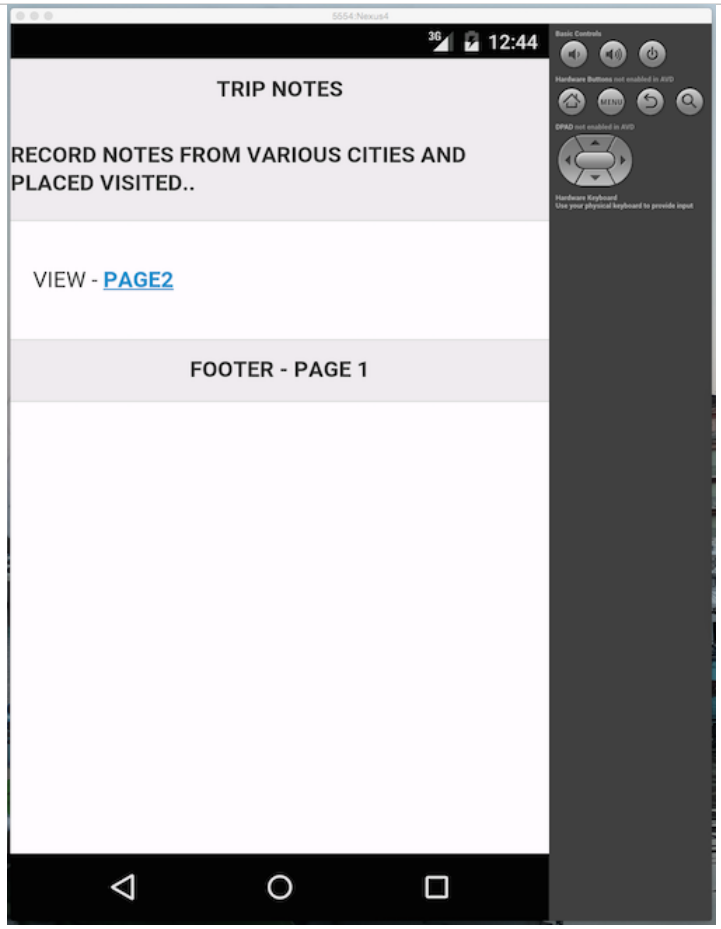
Cordova App - basics of development - part 6

jQuery Mobile - test transitions

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

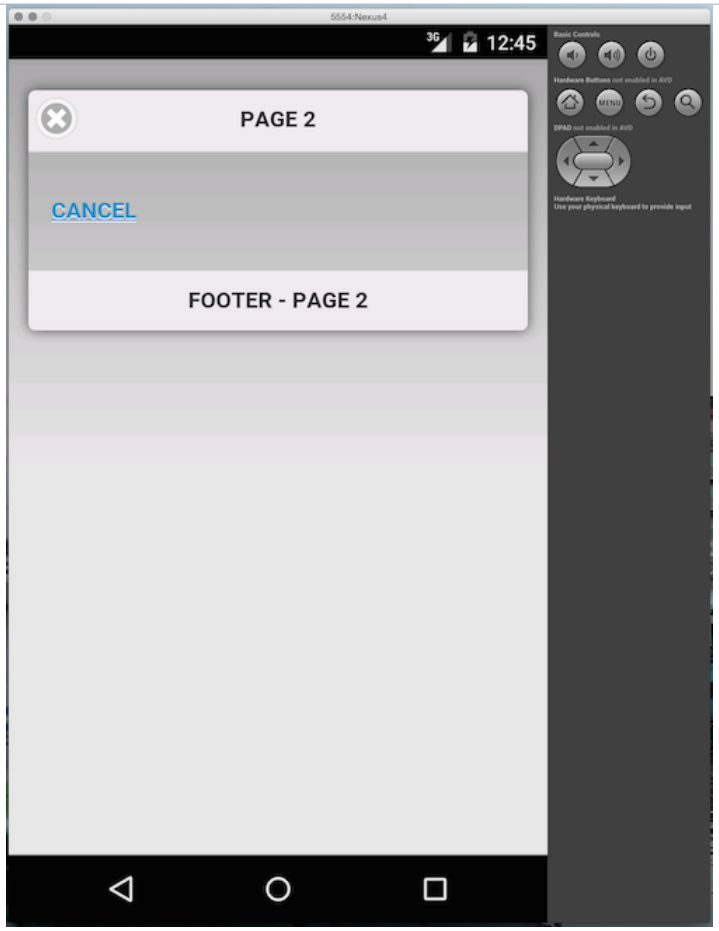
```
<!-- page1 -->
<div data-role="page" id="page1">
  <div data-role="header">
    <h3>trip notes</h3>
    <p>record notes from various cities and placed visited..</p>
  </div><!-- /header -->
  <div role="main" class="ui-content">
    <p>View - <a href="#page2" data-transition="slidedown">page2</a>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

- 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

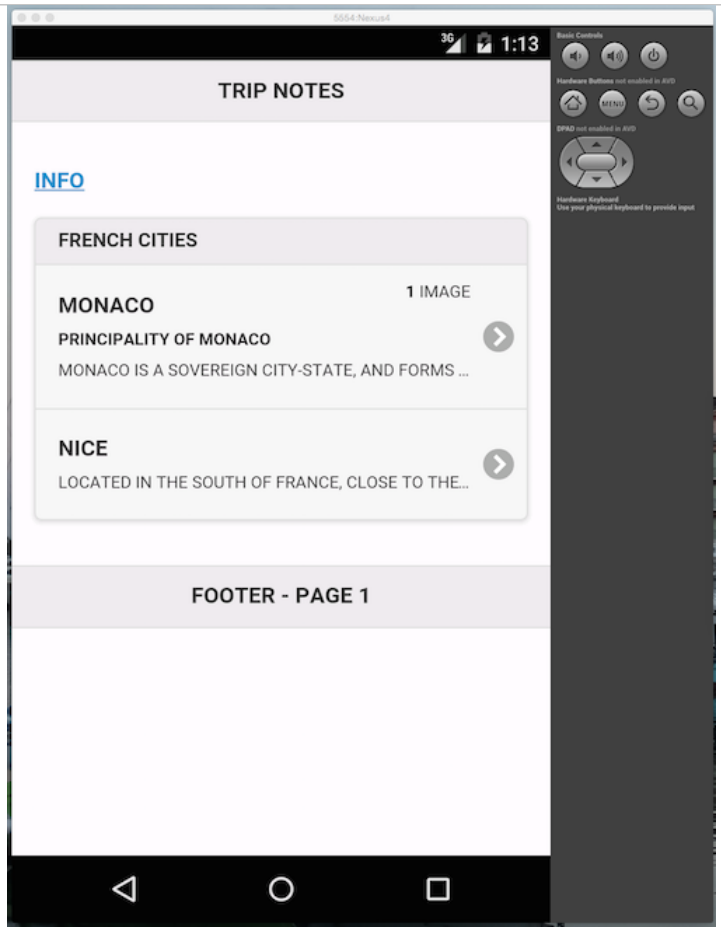
listviews - example

```
<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

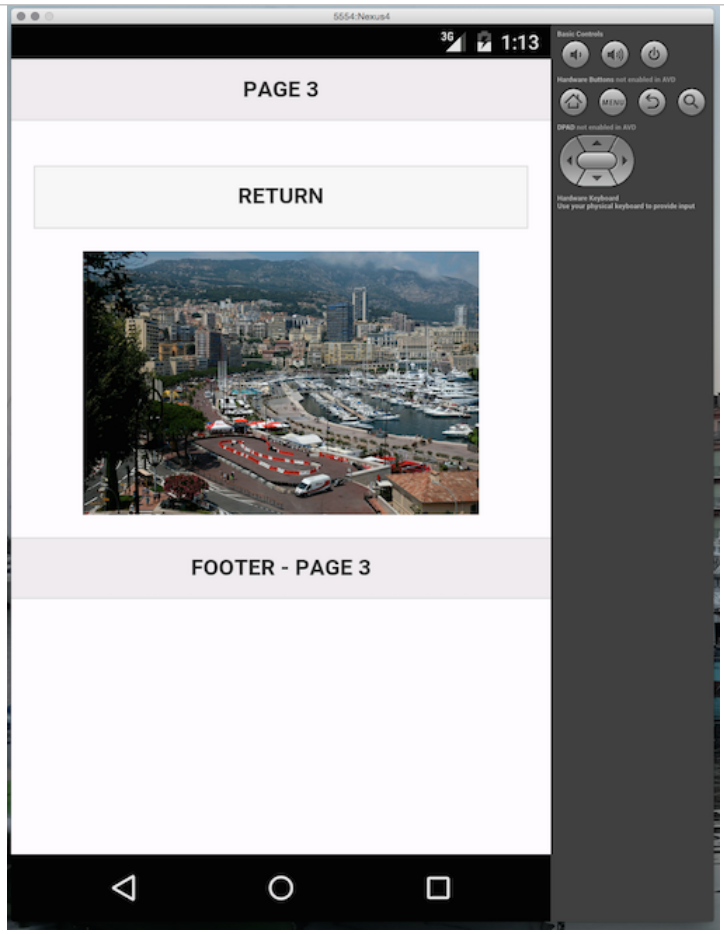
jQuery Mobile - add some organisation



Trip Notes - v0.06

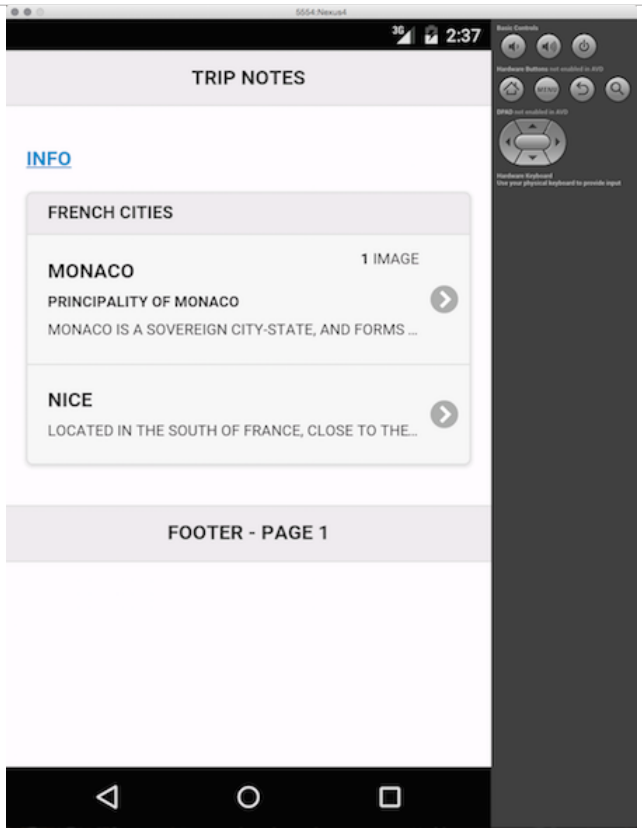
Cordova App - basic - part 8

jQuery Mobile - add some organisation



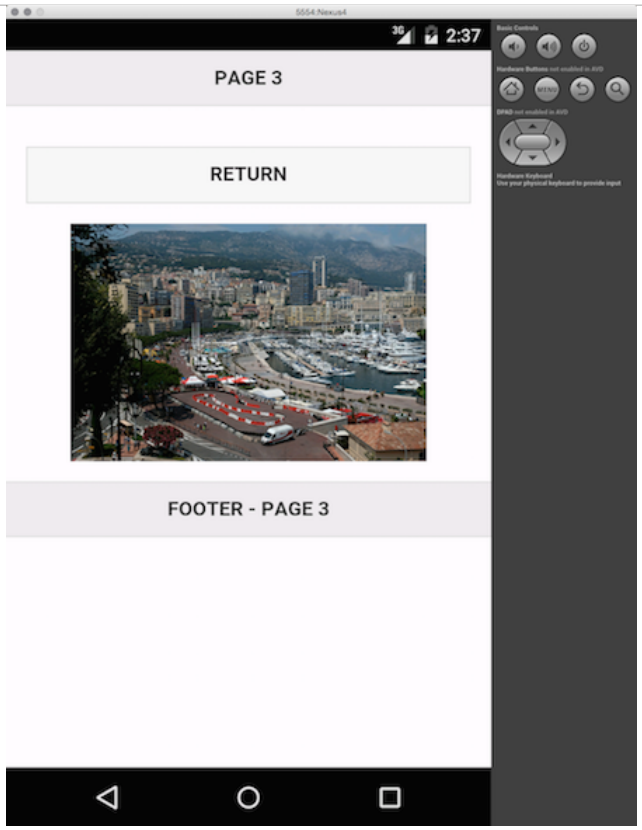
Trip Notes - v0.07

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

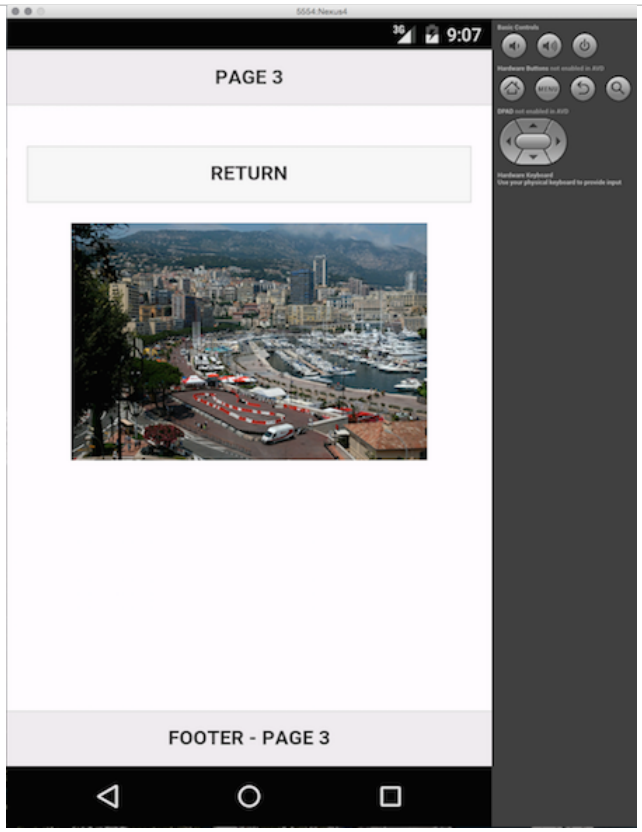
Cordova app - current design

- 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
```

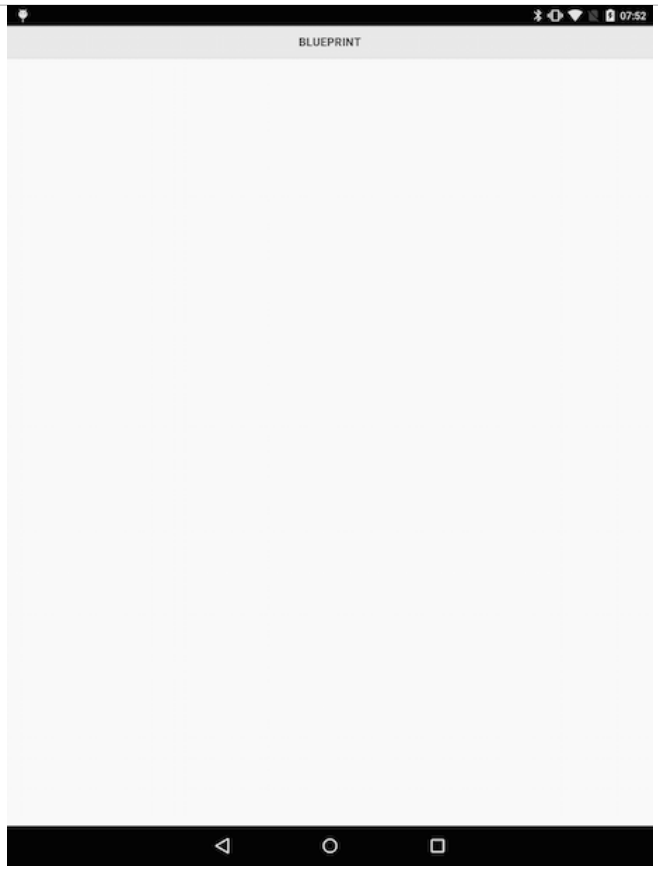
- add support for Android platform

```
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*

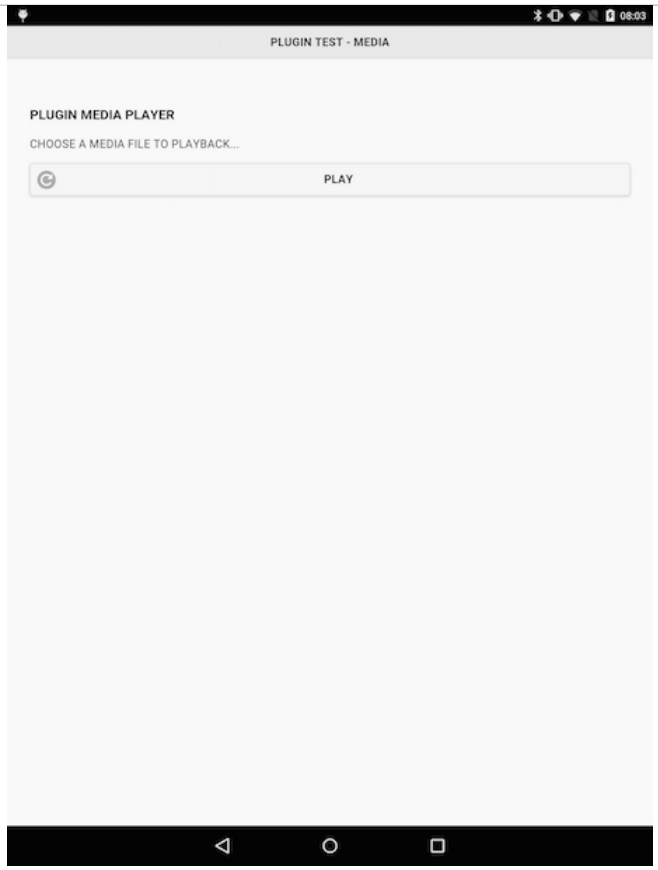
Cordova app - working with plugins - update `index.html`

- 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 &c. -->
      <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*

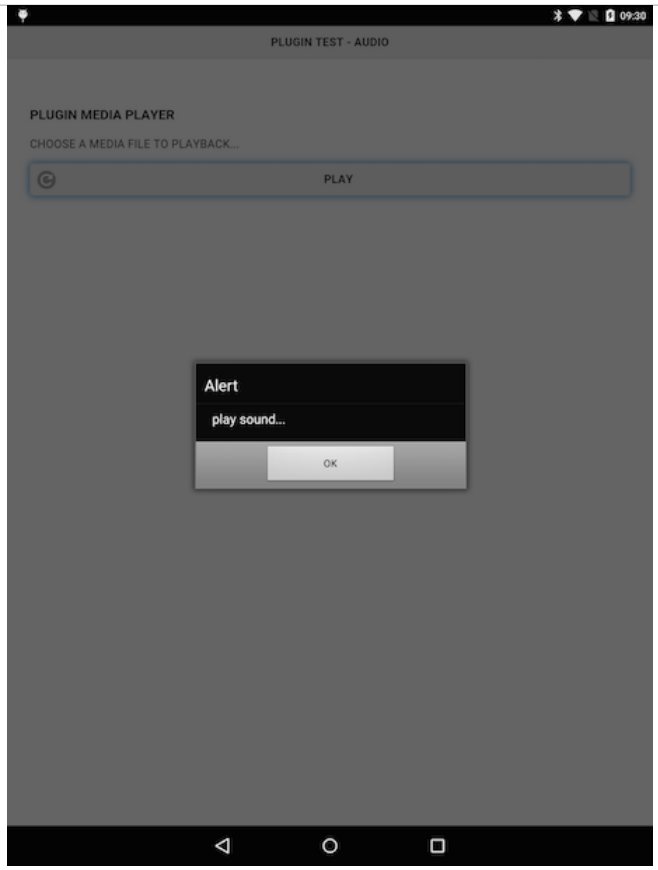
```
(function() {
    //check for page initialisation and #home
    $(document).on("pageinit", "#home", function(e) {
        //prevent any bound defaults
        e.preventDefault();

        //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

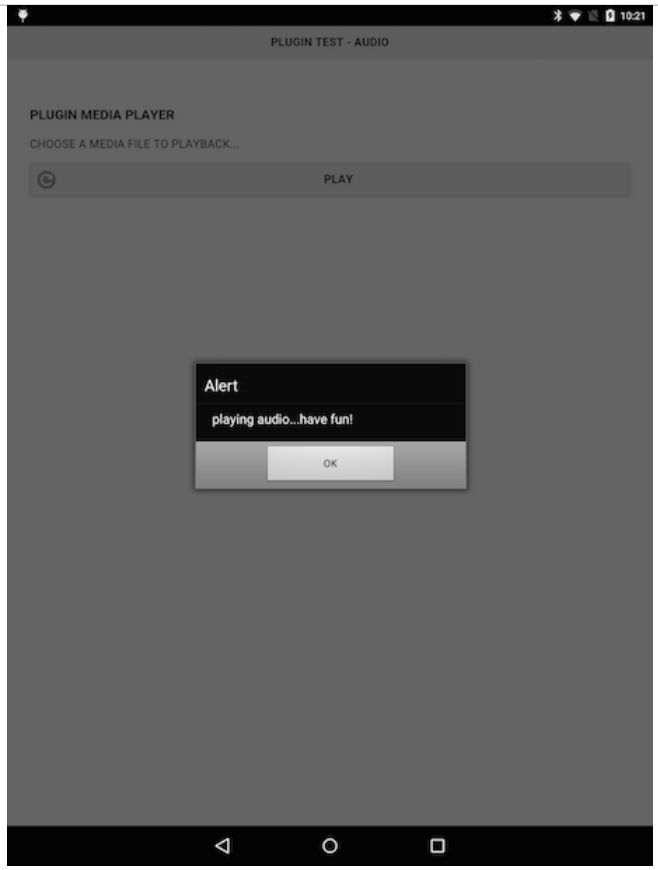
- now setup and tested the basic app logic
  - *added handlers for `deviceready` and clicking the audio playback button*
- update logic for the `#playAudio` button

```
//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

Cordova app - working with plugins - update media playback

- basic plugin test for media playback within an app
  - *user can play music in their app*
  - *user touch interaction with button*
  - *file loaded from local filesystem*
  - *device playback of selected audio file*
- leveraging native device functionality in app
  - *calling plugins for **device**, **file**, **media**...*
- basic app includes,
  - *user interaction in the UI*
  - *calls to the exposed JS API for the plugins*
  - *playback of audio by the native device*
- add further functionality
  - *stop, pause...*

Cordova app - working with plugins - stop button

- consider how to **stop**, **pause** playback
  - *e.g. UI interaction, timer, event...*
- app logic is very similar
  - *respond to **stop** event*
  - *call method*
  - ...
- methods for **stop**, **pause**, &c. available in plugin API

```
media.pause
media.stop
media.release
```

Cordova app - working with plugins - stop button - part I

- start to update our existing app by adding a **stop** button to the UI
  - *allow our user to simply tap a button to stop playback*

```
<p>Stop playback...</p>
<input type="button" id="stopAudio" data-icon="delete" value="Stop" />
```

- update initial JS logic for the app
  - *listen for tap event on **stop** button*
  - *then call the stop method on the **media** object*

```
//button - stop audio
$("#stopAudio").on("tap", function(e) {
    //stop audio logic
    e.preventDefault();
  //call custom method to handle stopping audio...
    stopAudio();
});
```

Cordova app - working with plugins - stop button - part 2

- add the logic for our custom method to stop the audio
  - *call as `stopAudio()`*

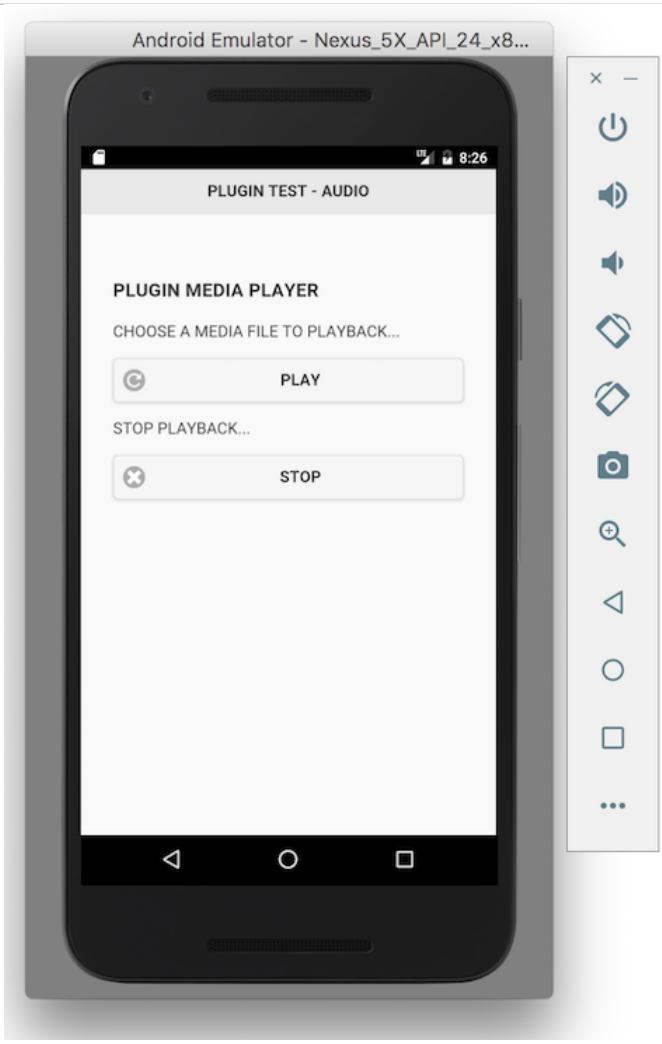
```
//stop audio file
function stopAudio() {
    //stop audio playback
    $audio.stop();
    //release audio - important for android resources...
    $audio.release();
    //just for testing
    alert("stop playing audio...& release!");
}
```

- logic still won't stop the audio playing
- issue is variable `$audio`
  - *currently restricted local scope to `playAudio()` method*
- initially alter scope of property for `$audio` itself
  - *now set in initial `onDeviceReady()` method*

```
function onDeviceReady() {
    //set initial properties
    var $audio;
...
}
```

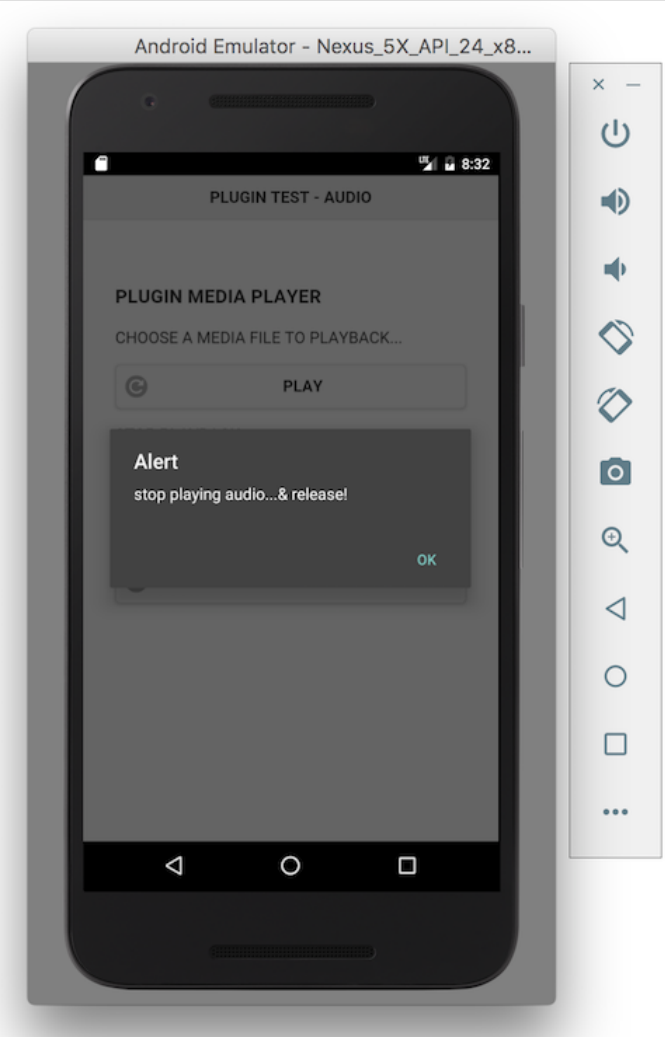
- logic will now stop audio playing
- call to `release()` method important for OS's audio resources
  - *particularly important to release unwanted resources on Android...*

Image - Cordova app - Plugin Test - stop audio playback



Cordova - Plugin Test - stop audio playback

Image - Cordova app - Plugin Test - stop audio playback 2



Cordova - Plugin Test - stop audio playback 2

Cordova app - working with plugins - pause button - part I

- follow similar pattern to add initial pause button to app's HTML

```
<p>Pause playback...</p>
<input type="button" id="pauseAudio" data-icon="bars" value="Pause" />
```

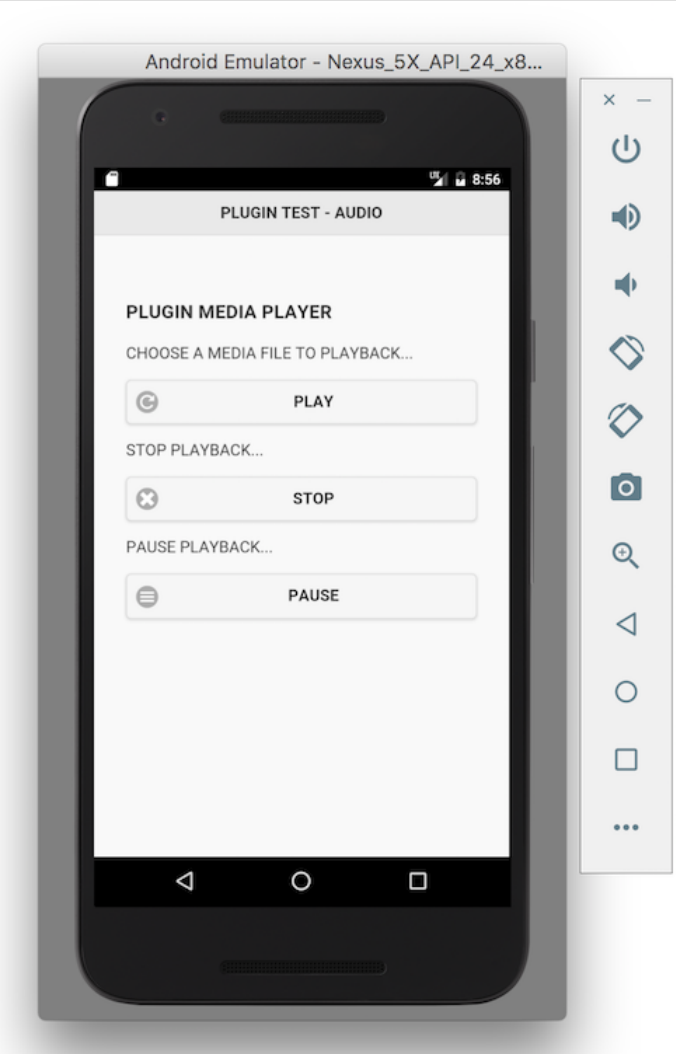
- then add basic listener for tap event on the pause button

```
//button - pause audio
$("#pauseAudio").on("tap", function(e) {
    //pause audio logic
    e.preventDefault();
    //call custom method to handle pausing audio...
    pauseAudio();
});
```

- then add our custom `pauseAudio()` method
  - *handles pausing of current media object*

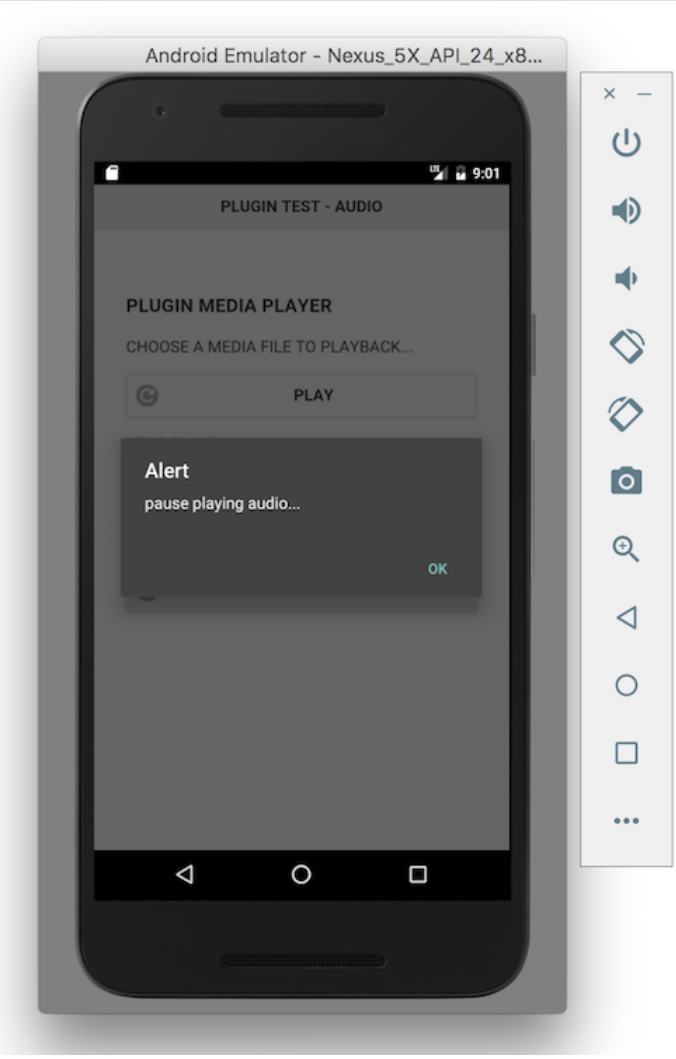
```
//pause audio file
function pauseAudio() {
    //pause audio playback
  $audio.pause();
}
```

Image - Cordova app - Plugin Test - pause audio playback



Cordova - Plugin Test - pause audio playback

Image - Cordova app - Plugin Test - pause audio playback 2

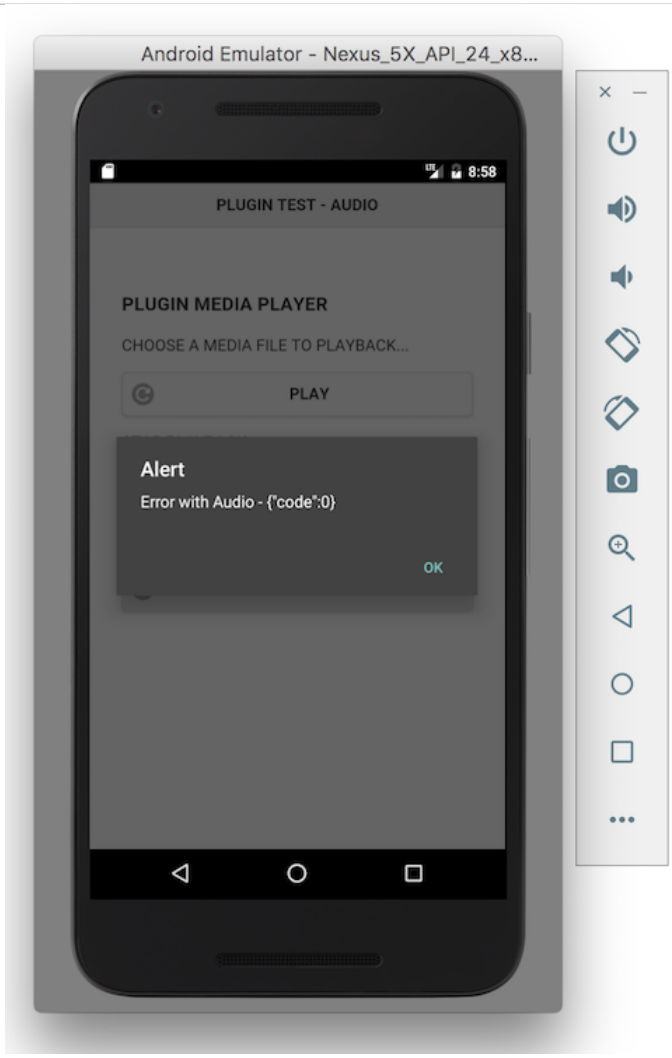


Cordova - Plugin Test - pause audio playback 2

Cordova app - working with plugins - pause button - part 2

- this logic works but it introduces issues and errors, e.g.
  - *start playback of audio and then pause*
  - *then touch play again*
  - *audio will restart from the start of the audio file*
  - *not ideal user experience...*
- an error will be thrown, e.g.
  - *press pause once, then twice...*
  - *error will be thrown for the call to the `pause()` method*

Image - Cordova app - Plugin Test - pause audio playback 3



Cordova - Plugin Test - pause audio playback 3

Cordova app - working with plugins - pause button - part 3

- we can monitor change in the playback with a simple property
  - *attached to scope for `onDeviceReady()` method*
  - *property available to `play()`, `pause()`, and `stop()` methods*

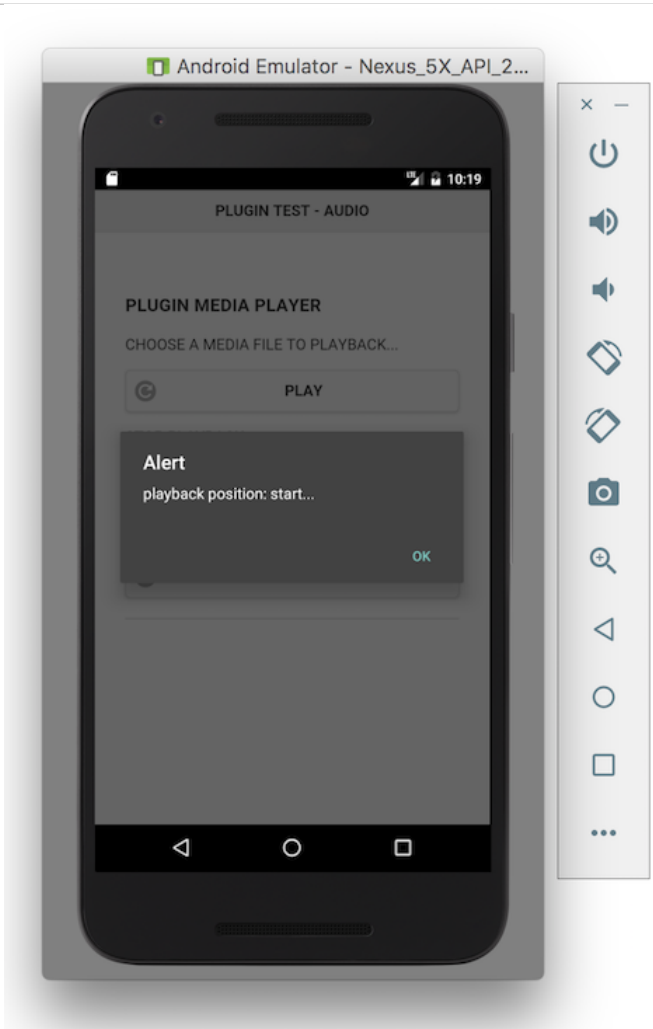
```
function onDeviceReady() {
  //set initial properties
    var $audio;
    var $audioPosn = 0;
...
}
```

- now have two properties we can monitor and update
  - *variable `$audioPosn` has been set to a default value of 0*
  - *we can check as we start to playback an audio file &c.*

```
//check current audio position
if ($audioPosn > 1) {
  $audio.play();
  alert("playback position: " + $audioPosn + " secs");
} else {
    $audio.play();
    alert("playback position: start...");
}
```

- also use property to output current playback position, reset for cancelling, &c.

Image - Cordova app - Plugin Test - update playback 1



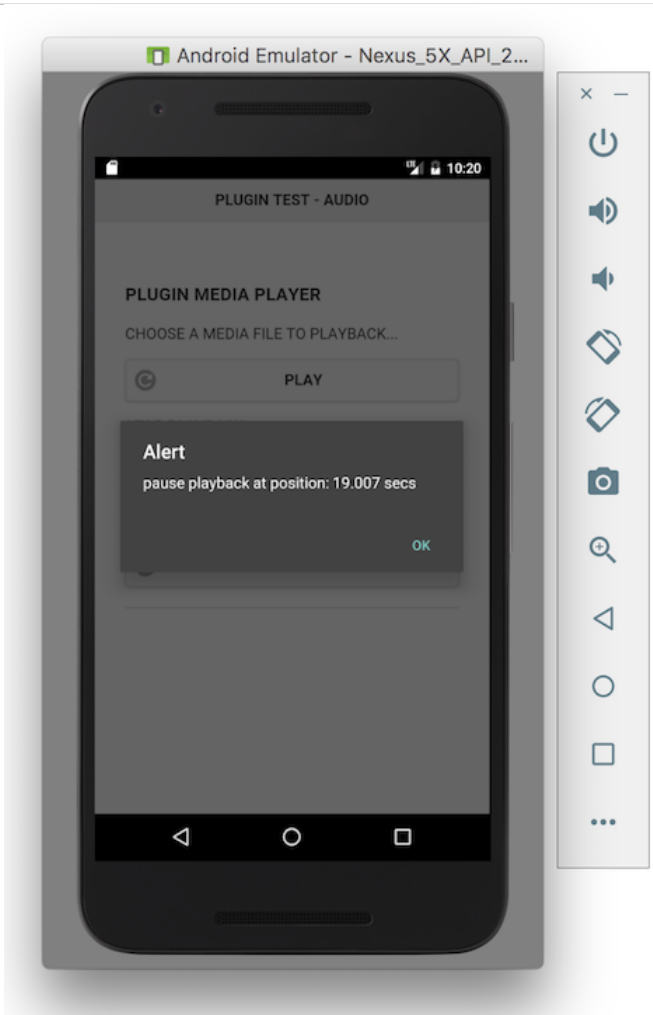
Cordova - Plugin Test - update playback

Cordova app - working with plugins - pause button - part 4

- pause a playing audio stream
  - *need to be able to get the current playback position for the audio file*
  - *then update our `$audioPosn` property.*
- check audio position in the `pauseAudio()` method
  - *use the `getCurrentPosition()` method*
  - *available on the `media` object...*

```
$audio.getCurrentPosition(
  // success callback
  function (position) {
    if (position > -1) {
            $audioPosn = position;
      alert("pause playback at position: " + position + " secs");
    }
  }, // error callback
     function (e) {
      ...
    }
);
```

Image - Cordova app - Plugin Test - update playback 2



Cordova - Plugin Test - update playback 2

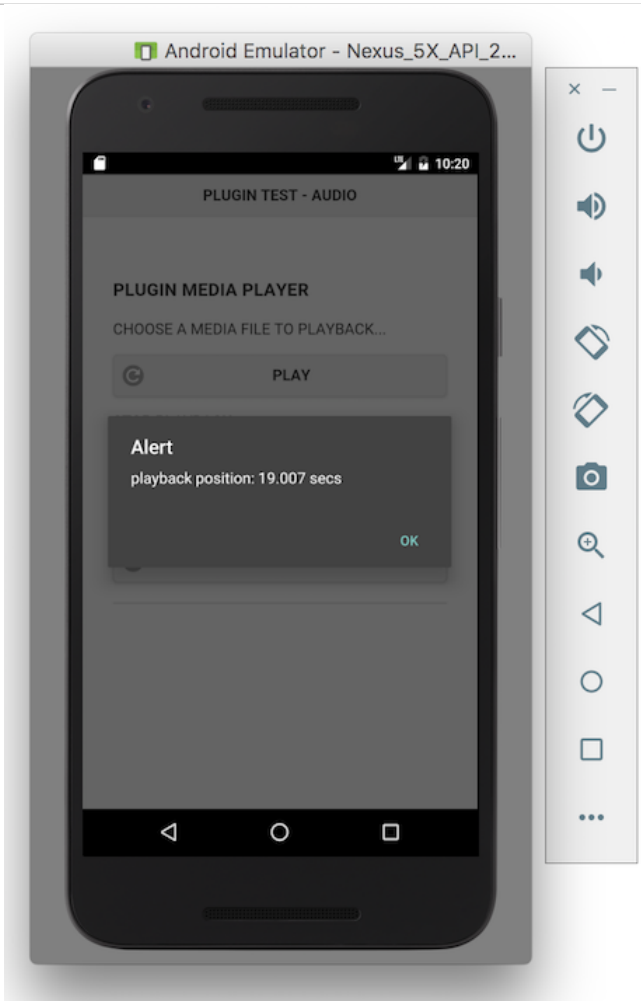
Cordova app - working with plugins - pause button - part 5

- we can now successfully pause our audio playback
  - *store value for current pause position in the audio stream*
- also need to update our audio playback
  - *need to check current position in audio stream*

```
//check current audio position
if ($audioPosn > 1) {
  $audio.seekTo($audioPosn*1000);
  $audio.play();
  alert("playback position: " + $audioPosn + " secs");
} else {
  $audio.play();
  alert("playback position: start...");
}
```

- we updated the `playAudio()` method to check value of `$audioPosn` property
- now use value to seek to current position in audio stream
  - *using `seekTo()` method exposed by `media` object itself...*
  - *method expects time in milliseconds*
  - *need to update value for our `$audioPosn` property, `$audioPosn*1000`*
- audio stream will now resume at correct position...

Image - Cordova app - Plugin Test - update playback 3



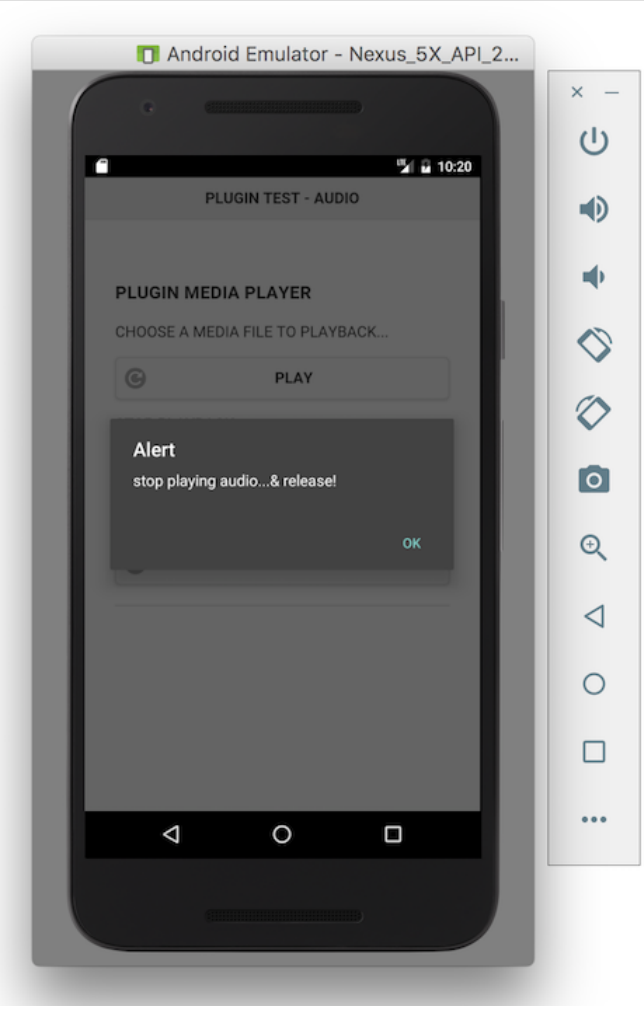
Cordova - Plugin Test - update playback 3

Cordova app - working with plugins - update stop button

- final touch for now, at least with the buttons
- need to update logic for app's **stop** button
- need to reset the value of the `$audioPosn` property
  - *if not, audio stream will always restart at set pause value*

```
//stop audio file
function stopAudio() {
  //stop audio playback
    $audio.stop();
    //reset $audioPosn
    $audioPosn = 0;
    //release audio - important for android resources...
    $audio.release();
    //just for testing
    alert("stop playing audio...& release!");
}
```

Image - Cordova app - Plugin Test - update playback 4



Cordova - Plugin Test - update playback 4

Cordova app - working with plugins - current playback position

- now seen how we can check the current position of a playing audio file
- many different options for outputting this value
  - *e.g. appending its value to the DOM, showing a dialogue, and so on...*
- how we use the value of this property is up to us as developers
  - *naturally informed by the requirements of the app*
- may only be necessary to use this value internally
  - *help with the app's logic*
- may need to output this result to the user

Cordova app - working with plugins - further considerations

A few updates and modifications for a media app

- update logic for app
  - *checks for event order, property values, &c.*
- indicate playback has started
  - ***without*** *alerts...*
- update state of buttons in response to app state
  - *highlights, colour updates...*
- inactive buttons and controls when not needed
  - *update state of buttons...*
- grouping of buttons to represent media player
  - *add correct icons, playback options...*
- metadata for audio file
  - *title, artist, length of track...*
- image for track playing
  - *thumbnail for track, album...*
- track description
- notification for track playing
- persist track data and choice in cache for reload...
- ...

Cordova app - working with plugins - add splashscreen

- add support for splashscreens in Cordova
  - *install splashscreen plugin in project*

```
cordova plugin add cordova-plugin-splashscreen
```

- then we need to return to our `config.xml` file
  - *set different splashscreens for different supported platforms*
  - *specify different images to use for given screen resolutions*
- Android example,

```
<platform name="android">
  <!-- splashscreens - you can use any density that exists in the Android project -->
  <!-- landscape splashscreens -->
  <splash src="res/screen/android/splash-land-hdpi.png" density="land-hdpi"/>
  <splash src="res/screen/android/splash-land-ldpi.png" density="land-ldpi"/>
  <splash src="res/screen/android/splash-land-mdpi.png" density="land-mdpi"/>
  <splash src="res/screen/android/splash-land-xhdpi.png" density="land-xhdpi"/>
  <!-- portrait splashscreens -->
  <splash src="res/screen/android/splash-port-hdpi.png" density="port-hdpi"/>
  <splash src="res/screen/android/splash-port-ldpi.png" density="port-ldpi"/>
  <splash src="res/screen/android/splash-port-mdpi.png" density="port-mdpi"/>
  <splash src="res/screen/android/splash-port-xhdpi.png" density="port-xhdpi"/>
</platform>
```

- specifying different images for each screen density
  - *then specify for portrait and landscape aspect ratios*
- URL for the `src` attribute is relative to the project's root directory
  - *not the customary www*

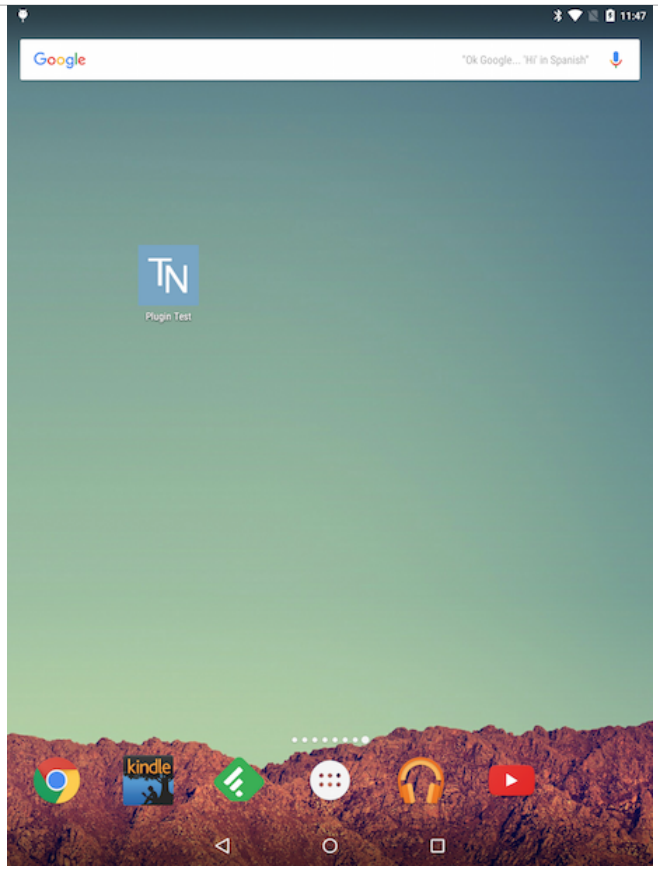
Cordova app - working with plugins - add an app icon

- also set our own app's icon
  - *again in the `config.xml` setting for the application*

```
<platform name="android">
  <icon src="res/icon/android/ldpi.png" density="ldpi" />
  <icon src="res/icon/android/icon/mdpi.png" density="mdpi" />
  <icon src="res/icon/android/icon/hdpi.png" density="hdpi" />
  <icon src="res/icon/android/icon/xhdpi.png" density="xhdpi" />
</platform>
```

- again, we can target specific platforms
  - *useful way to handle different screen resolutions and densities*
- icon's URL is specified relative to the project's root directory

Image - Cordova app - Plugin Test 1 - getting started



Cordova - Plugin Test - custom icon

Cordova app - working with plugins - Android icon sizes for launcher

Density	Launcher icon size
ldpi	36 x 36 px
mdpi	48 x 48 px
hdpi	72 x 72 px
xhdpi	96 x 96 px

and so on...

Cordova app - test with local tools

- default testing options with Cordova CLI include
  - *`emulate` and `run`*
- other options available as well
- Apache Ripple - install using NPM

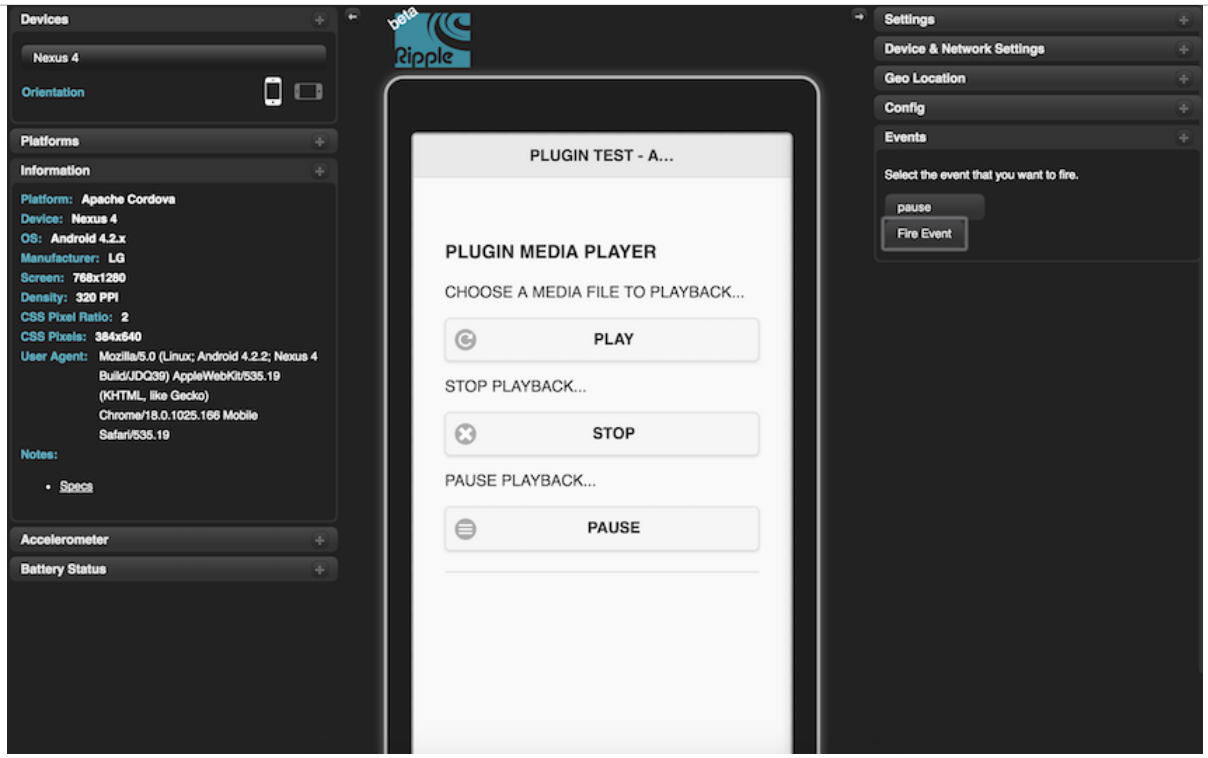
```
npm install -g ripple-emulator
```

- then, `cd` to working directory of Cordova app and run the following command

```
ripple emulate
```

- not the most up to date emulator, but useful for quick UI and interaction testing
- Genymotion - target at Android development, testing, and provision
  - *professional development and testing options available*
  - *further details at https://www.genymotion.com*

Image - Cordova app - test with local tools - Apache Ripple



Cordova app - test with local tools - Apache Ripple

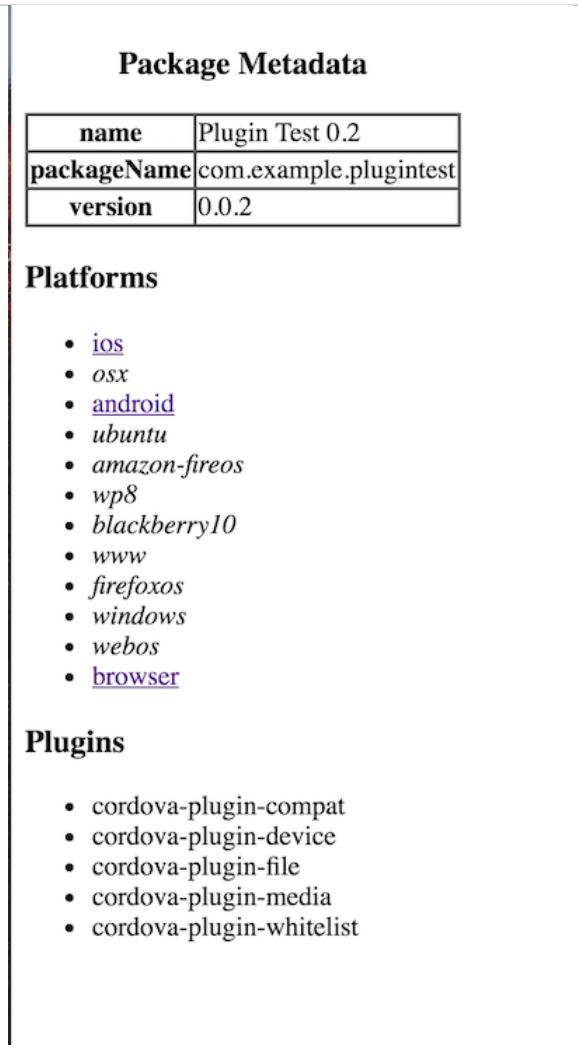
Cordova app - test with local tools - `serve`

- Cordova also provides the option to **serve** a current app
- `serve` as self-hosted site for testing

```
cordova serve
```

- start a local static file server at `http://localhost:8000`
  - *then navigate to a given platform's directory*
  - *and the associated project UI and build*
  - *useful for UI testing and quick development*

Image - Cordova app - test with local server - `serve`



Package Metadata

name	Plugin Test 0.2
packageName	com.example.plugintest
version	0.0.2

Platforms

- ios
- *osx*
- android
- *ubuntu*
- *amazon-fireos*
- *wp8*
- *blackberry10*
- *www*
- *firefoxos*
- *windows*
- *webos*
- browser

Plugins

- cordova-plugin-compat
- cordova-plugin-device
- cordova-plugin-file
- cordova-plugin-media
- cordova-plugin-whitelist

Cordova app - test with local server - serve

Cordova app - test with local tools - Chrome browser and device

- test and develop Android applications with **devices** on Chrome browser
- after running our app on a connected device, e.g.

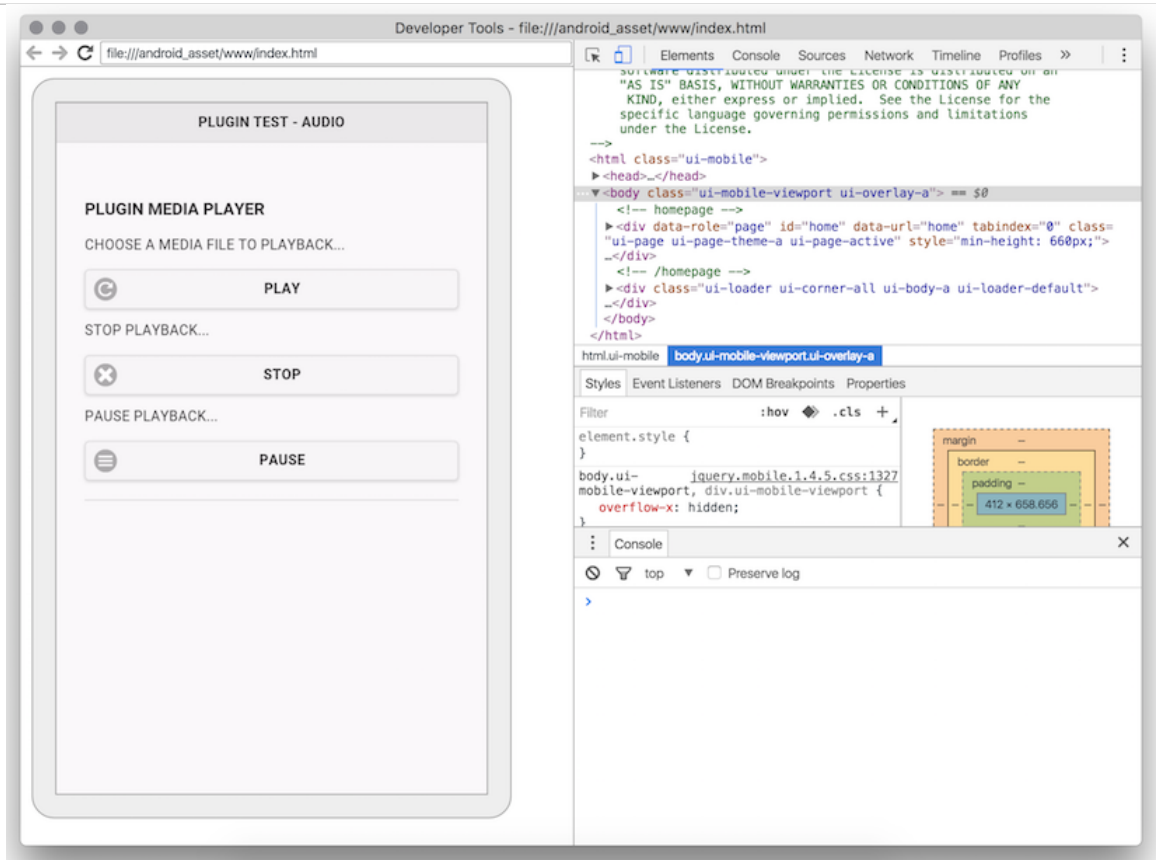
```
cordova run android
```

- inspect the app using Chrome's developer tools at the following URL,

```
html    chrome://inspect/#devices
```

- then select the option to `inspect` a connected device
- shows window with the standard Chrome developer tools and options
  - *inspect the DOM, JS console, styles, and so on...*
  - *use inspect option to control, navigate, and interact with our running app*

Image - Cordova app - test with local server - Chrome



Cordova app - test with local server - Chrome

Cordova app - test with Browser platform

- Cordova recently added a **Browser** platform option
- use to create a quasi-test environment for our apps
- install browser support as a standard platform

```
cordova platform add browser
```

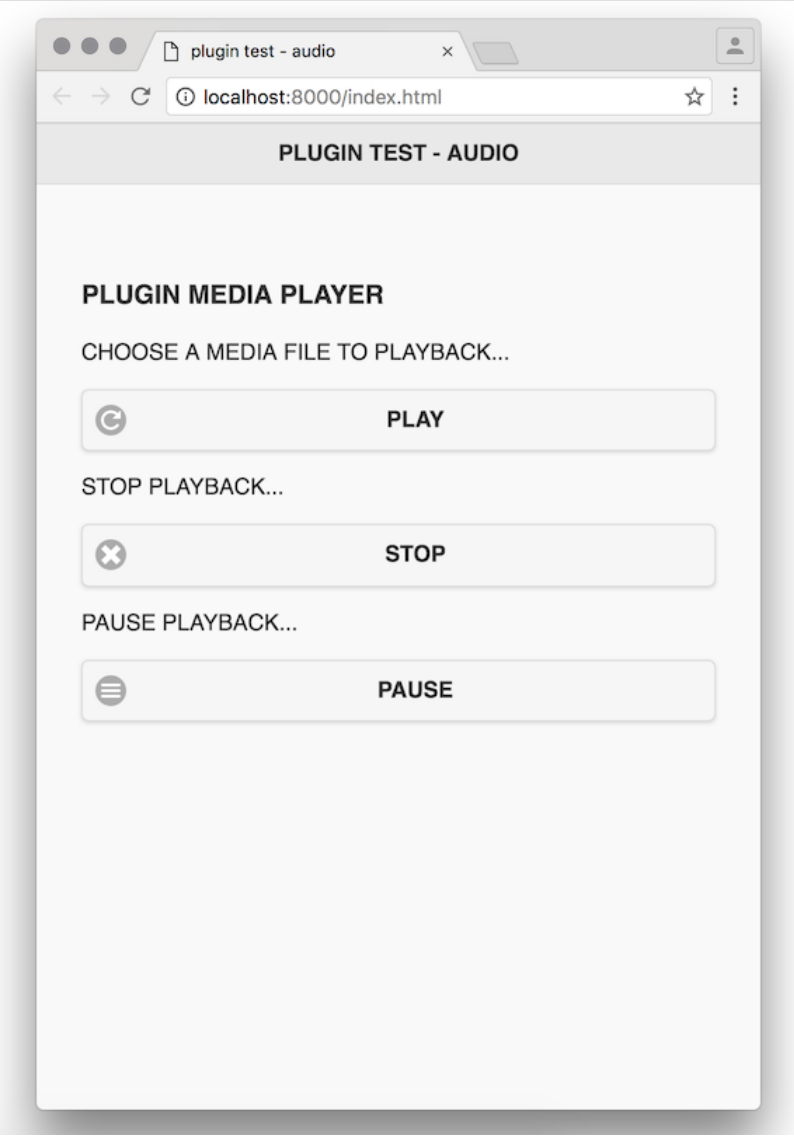
- load our app into the browser using the following command,

```
cordova run browser
```

- platform will be useful for testing UI design and development
- many of the plugins are supported as well
  - *e.g. camera*

n.b. *other options better for testing development of custom or OS level Android or iOS features...*

Image - Cordova app - test with browser platform



Cordova app - browser platform

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 API
- config.xml
- plugins
- plugin - device
- plugin - file
- plugin - media
- plugin - Splashscreen
- HTML5
- HTML5 File API
- jQuery Mobile
- API documentation
- Browser Support
- Pagecontainer widget