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

Fall Semester 2018 - Week 3

Dr Nick Hayward

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*

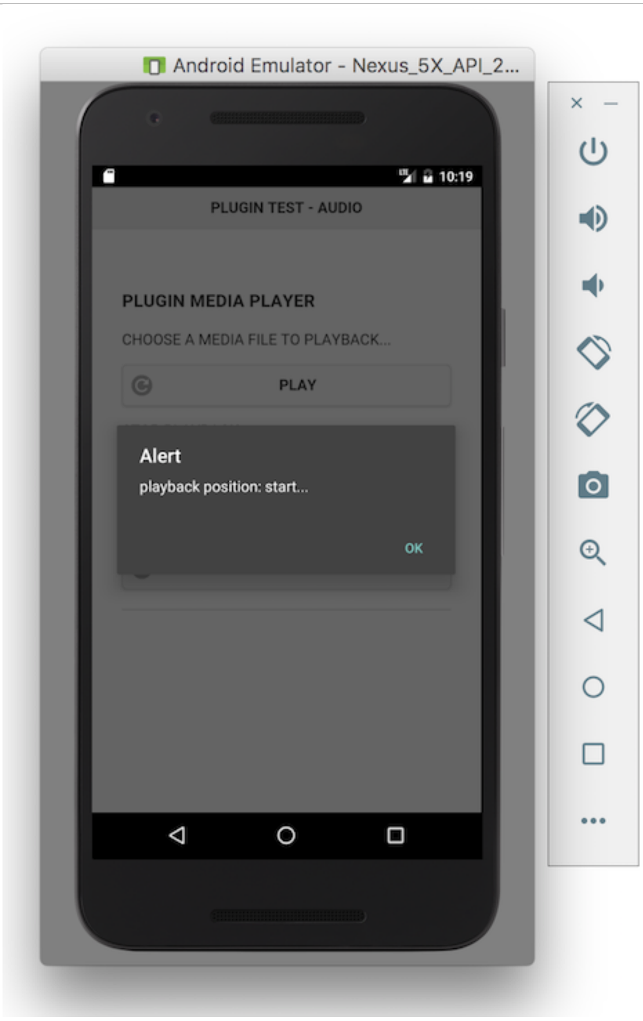
```
function onDeviceReady() {
  //set initial properties
    var $audio;
    var $audioPosn = 0;
...
}
```

- 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


Android Emulator - Nexus_5X_API_2...
10:19
PLUGIN TEST - AUDIO
PLUGIN MEDIA PLAYER
CHOOSE A MEDIA FILE TO PLAYBACK...
PLAY

Alert

playback position: start...

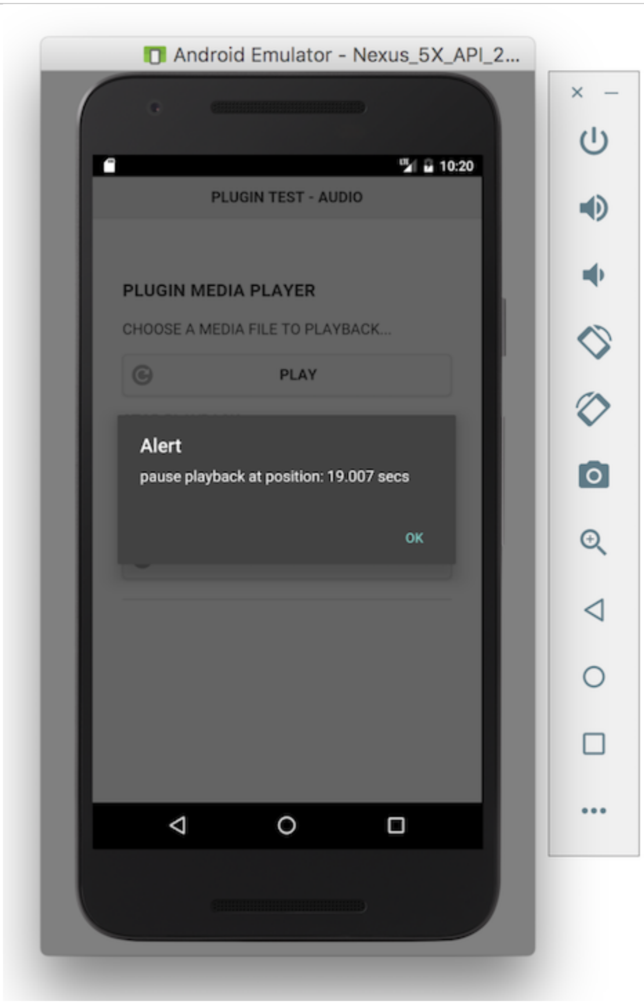
Cordova - Plugin Test - update playback

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


Android Emulator - Nexus_5X_API_2...
10:20
PLUGIN TEST - AUDIO
PLUGIN MEDIA PLAYER
CHOOSE A MEDIA FILE TO PLAYBACK...
PLAY
OK

Alert

pause playback at position: 19.007 secs

Cordova - Plugin Test - update playback 2

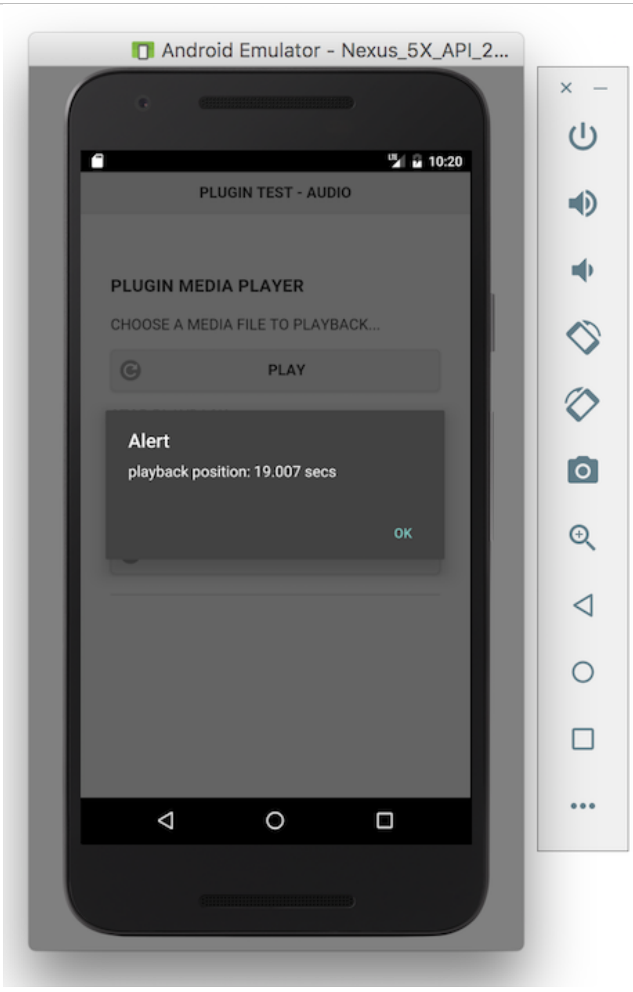
Cordova app - working with plugins - pause button - part 5

- 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


Android Emulator - Nexus_5X_API_2...
10:20
PLUGIN TEST - AUDIO
PLUGIN MEDIA PLAYER
CHOOSE A MEDIA FILE TO PLAYBACK...
PLAY

Alert

playback position: 19.007 secs

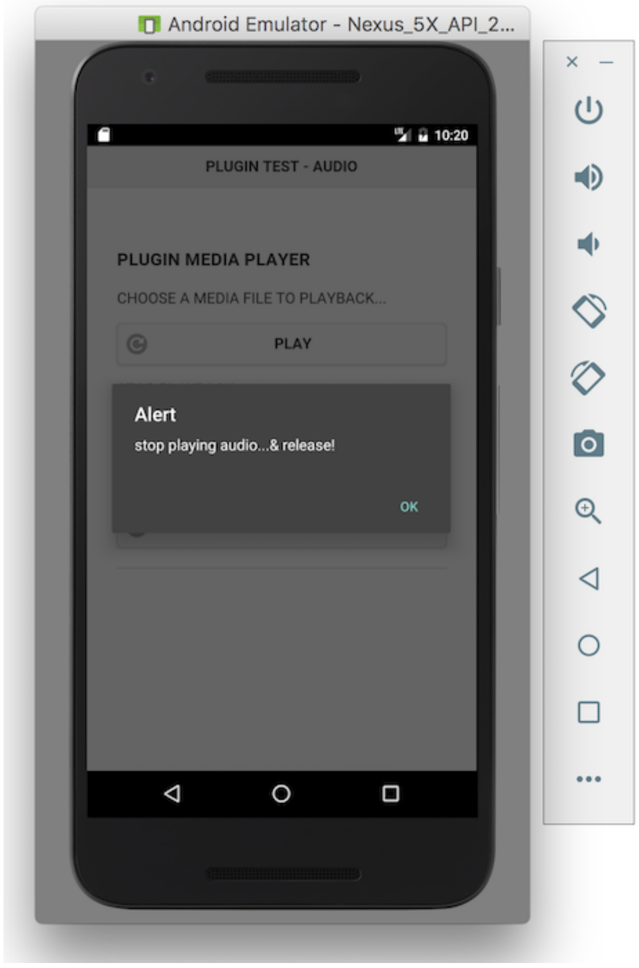
Cordova - Plugin Test - update playback 3

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


Android Emulator - Nexus_5X_API_2...
10:20
PLUGIN TEST - AUDIO
PLUGIN MEDIA PLAYER
CHOOSE A MEDIA FILE TO PLAYBACK...
PLAY
Alert
OK

stop playing audio...& release!

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`

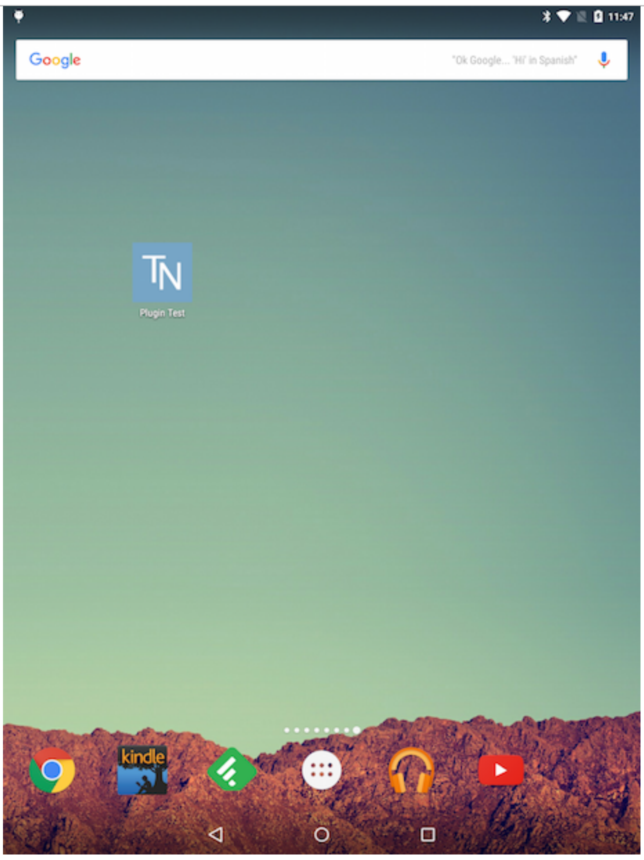
Cordova app - working with plugins - add an app icon

- 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

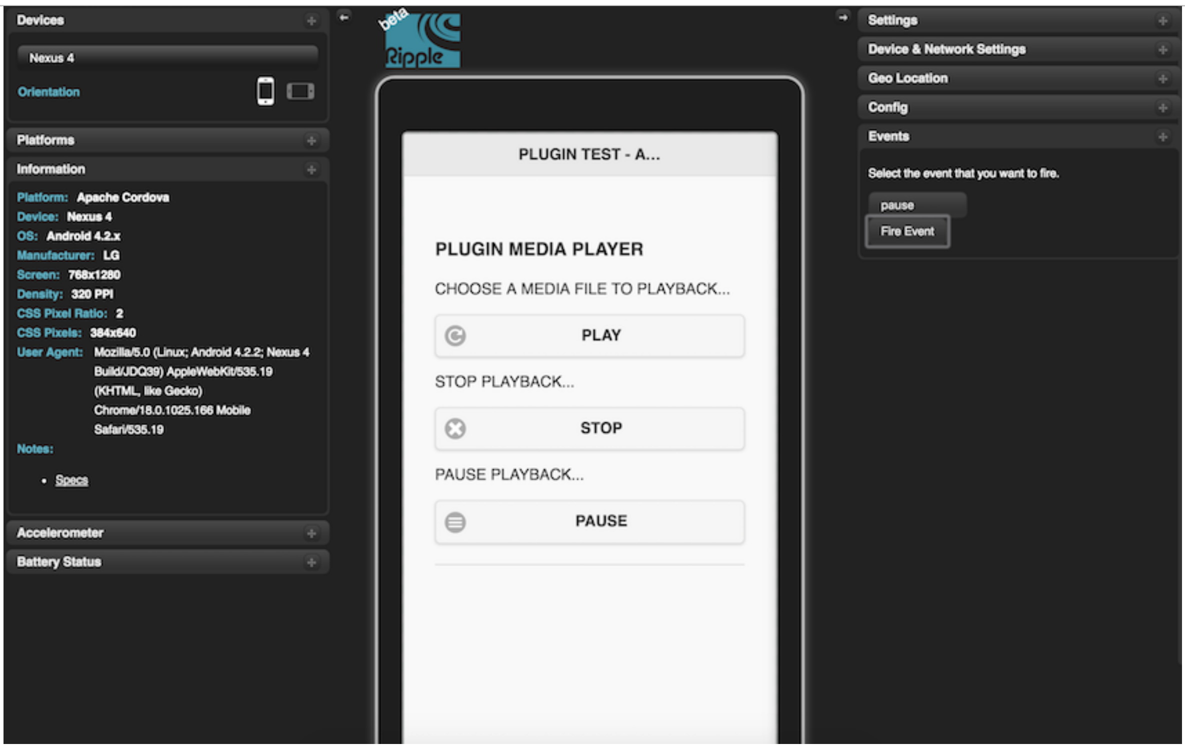
Density	Launcher icon size
ldpi	36 x 36 px
mdpi	48 x 48 px
hdpi	72 x 72 px
xhdpi	96 x 96 px

and so on...

Cordova app - test with local tools

- default testing options with Cordova CLI include
  - *`emulate` and `run`*
- many options available as well...
- e.g. Cordova testing tools
- 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



Platform: Apache Cordova
Device: Nexus 4
OS: Android 4.2.x
Manufacturer: LG
Screen: 768x1280
Density: 320 PPI
CSS Pixel Ratio: 2
CSS Pixels: 384x640
User Agent: Mozilla/5.0 (Linux; Android 4.2.2; Nexus 4 Build/JDQ39) AppleWebKit/535.19 (KHTML, like Gecko) Chrome/18.0.1025.166 Mobile Safari/535.19
Notes:

- Specs

PLUGIN MEDIA PLAYER

CHOOSE A MEDIA FILE TO PLAYBACK...

STOP PLAYBACK...

PAUSE PLAYBACK...

Select the event that you want to fire.

Cordova app - test with local tools - Apache Ripple

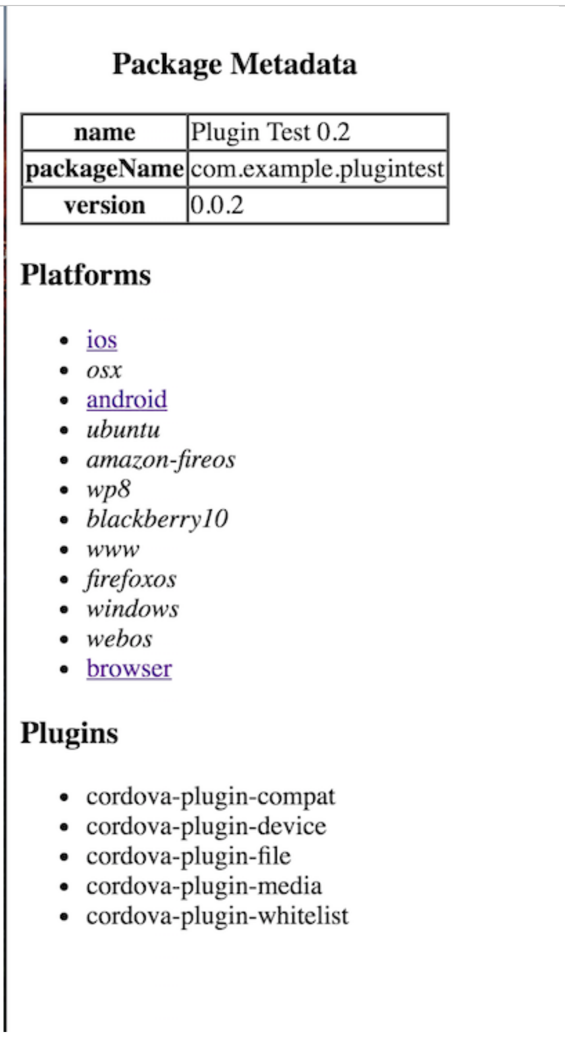
Cordova app - test with local tools - `serve`

- 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.

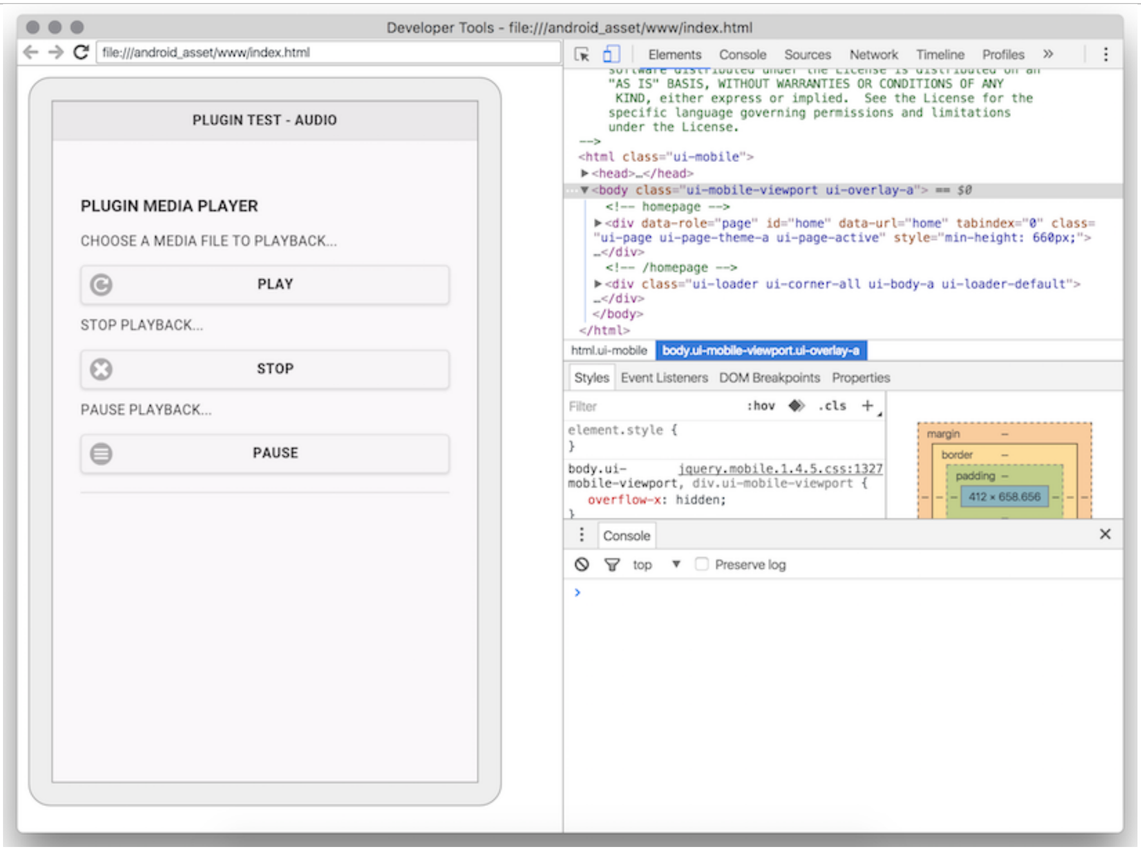
```
cordova run android
```

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

```
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



```
    "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
     KIND, either express or implied.  See the License for the
    specific language governing permissions and limitations
    under the License.
-->
<html class="ui-mobile">
▶<head>…</head>
▼<body class="ui-mobile-viewport ui-overlay-a"> == $0
    <!-- homepage -->
  ▶<div data-role="page" id="home" data-url="home" tabindex="0" class=
  "ui-page ui-page-theme-a ui-page-active" style="min-height: 660px;">
  …</div>
    <!-- /homepage -->
  ▶<div class="ui-loader ui-corner-all ui-body-a ui-loader-default">
  …</div>
  </body>
</html>
```

```
element.style {
}
body.ui-        jquery.mobile.1.4.5.css:1327
mobile-viewport, div.ui-mobile-viewport {
  overflow-x: hidden;
}
```

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
```

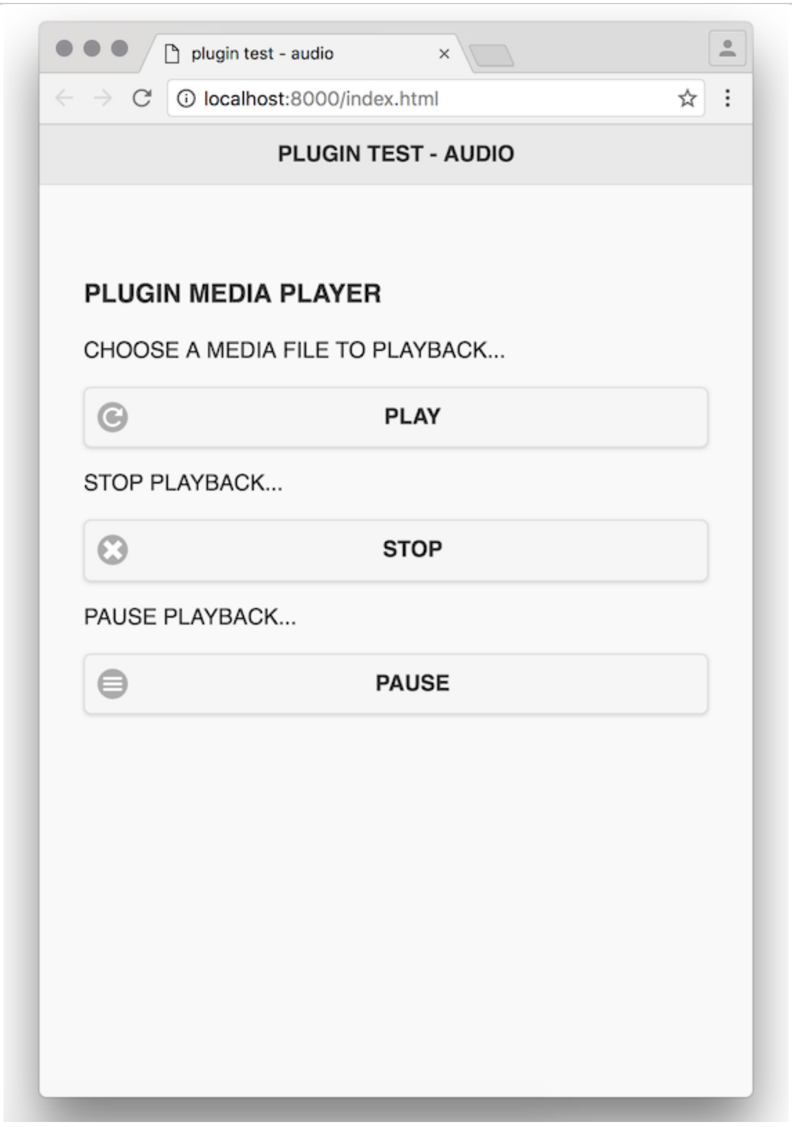
- load our app into the browser using the following command,

```
cordova run browser
```

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

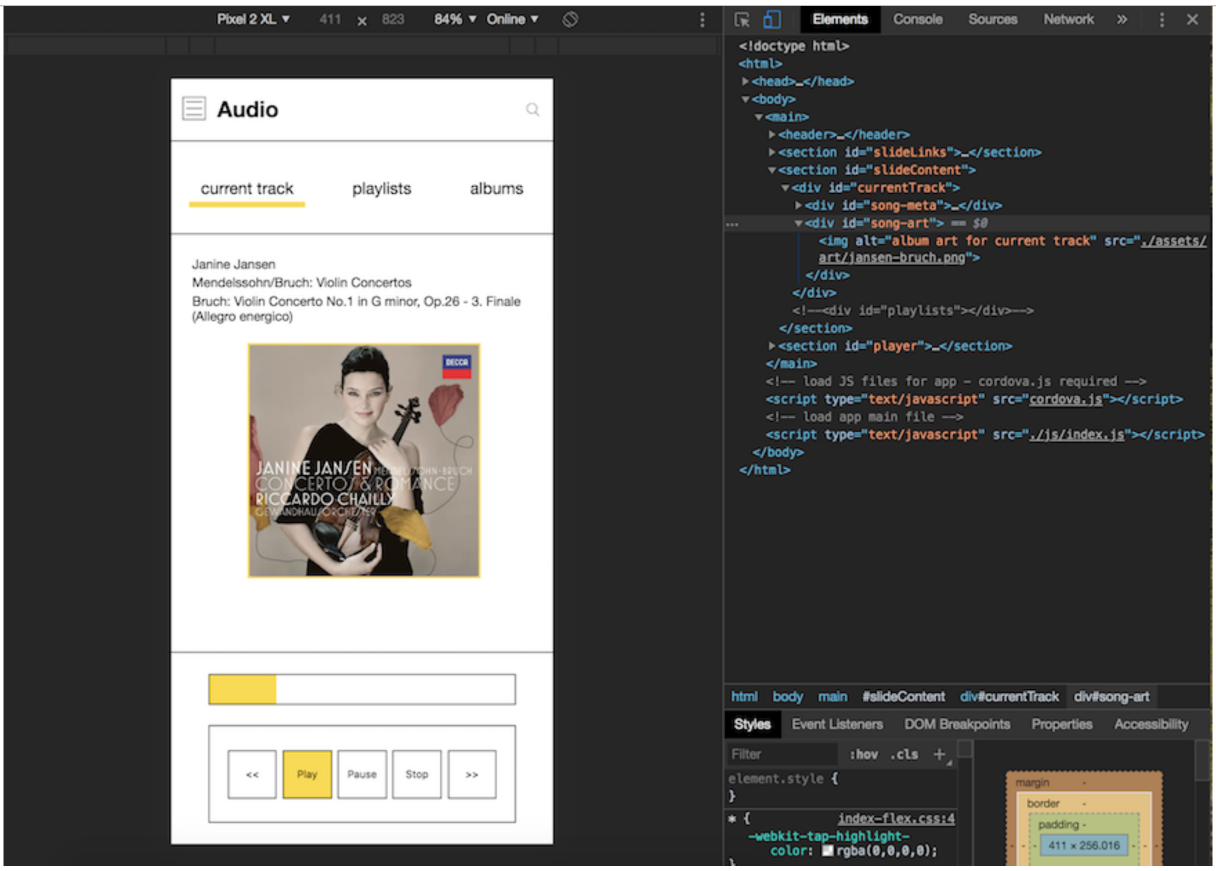
n.b. *other options better for testing development of custom or OS level Android or iOS features...*

Image - Cordova app - test with browser platform



Cordova app - browser platform

Image - Cordova app - test with browser platform - Chrome Dev Tools



Pixel 2 XL ▾ 411 × 823 84% ▾ Online ▾
Audio
current track
playlists
albums
Janine Jansen
Mendelssohn/Bruch: Violin Concertos
Bruch: Violin Concerto No.1 in G minor, Op.26 - 3. Finale (Allegro energico)
<<
Play
Pause
Stop
>>
Elements
Console
Sources
Network
html body main #slideContent div#currentTrack div#song-art
Styles
Event Listeners
DOM Breakpoints
Properties
Accessibility
Filter
:hov .cls +
margin -
border -
padding -
411 × 256.016

```
<!doctype html>
<html>
▸<head>…</head>
▾<body>
  ▾<main>
    ▸<header>…</header>
    ▸<section id="slideLinks">…</section>
    ▾<section id="slideContent">
      ▾<div id="currentTrack">
        ▸<div id="song-meta">…</div>
...     ▾<div id="song-art"> == $0
          <img alt="album art for current track" src="./assets/
          art/jansen-bruch.png">
         </div>
       </div>
       <!--<div id="playlists"></div>-->
     </section>
    ▸<section id="player">…</section>
   </main>
   <!-- load JS files for app - cordova.js required -->
   <script type="text/javascript" src="cordova.js"></script>
   <!-- load app main file -->
   <script type="text/javascript" src="./js/index.js"></script>
  </body>
</html>
```

```
element.style {
}
* {                         index-flex.css:4
  -webkit-tap-highlight-
     color: rgba(0,0,0,0);
}
```

Cordova app - browser platform - Chrome DEV tools

Cordova app - testing and automation with Microsoft's App Center

- App Center
- AppCenter Testing

Cordova app - automation with FastLane

- Fastlane - Overview

Mobile Design - Touch Events & Interaction

Fun exercise

Choose one of the following app types,

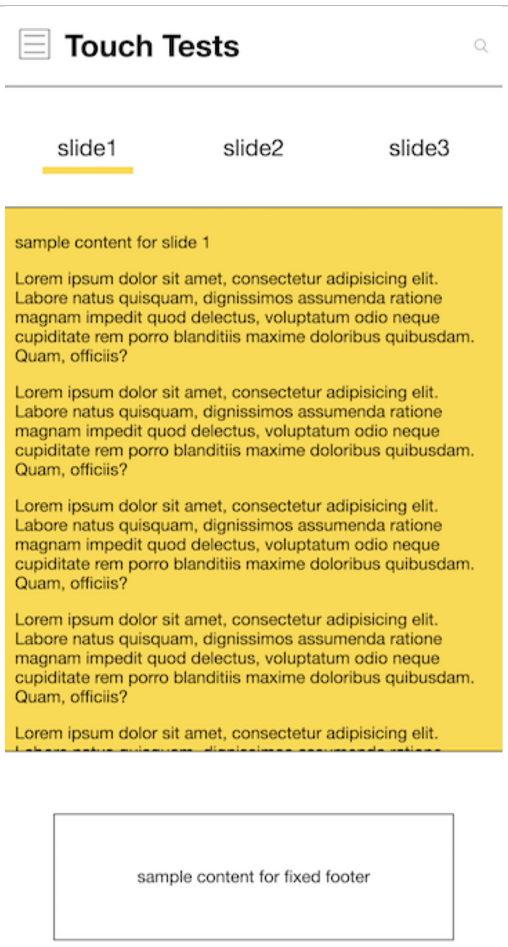
- mobile game - genre &c. is your choice...
- media app - audio or video (or both) playback options...
- fitness and geolocation app - track exercise, find locations &c.

Then, consider the following

- required **touch** events within this app
- role of these events relative to executed action
  - *i.e. what is the expected result of a touch event in the UI*
  - *consider logic and code execution...*
- UX options associated to a given touch event
  - *i.e. what is updated or added in the UI design*
  - *e.g. highlights, animations &c.*

~ 10 minutes

Image - Mobile Design - Touch Events & Interaction



sample content for slide 1

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Labore natus quisquam, dignissimos assumenda ratione magnam impedit quod delectus, voluptatum odio neque cupiditate rem porro blanditiis maxime doloribus quibusdam. Quam, officiis?

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Labore natus quisquam, dignissimos assumenda ratione magnam impedit quod delectus, voluptatum odio neque cupiditate rem porro blanditiis maxime doloribus quibusdam. Quam, officiis?

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Labore natus quisquam, dignissimos assumenda ratione magnam impedit quod delectus, voluptatum odio neque cupiditate rem porro blanditiis maxime doloribus quibusdam. Quam, officiis?

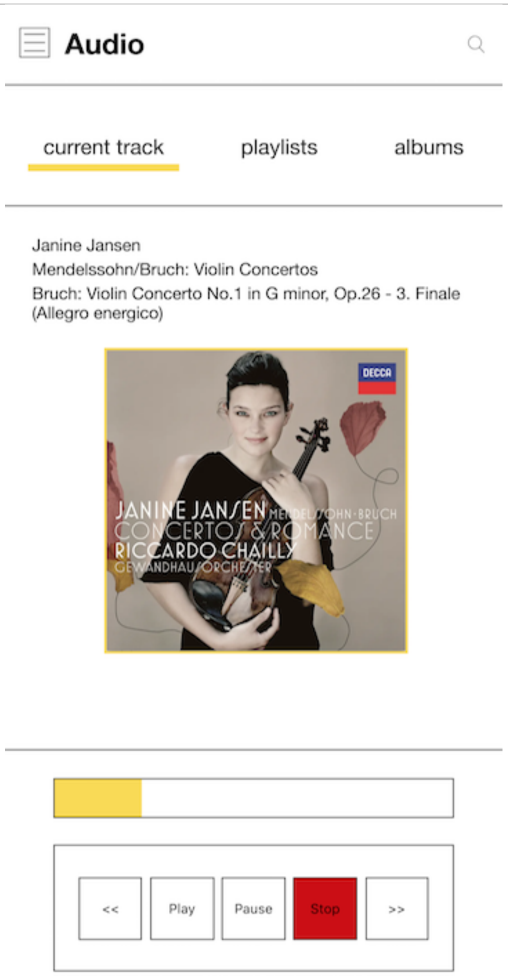
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Labore natus quisquam, dignissimos assumenda ratione magnam impedit quod delectus, voluptatum odio neque cupiditate rem porro blanditiis maxime doloribus quibusdam. Quam, officiis?

Lorem ipsum dolor sit amet, consectetur adipisicing elit.

sample content for fixed footer

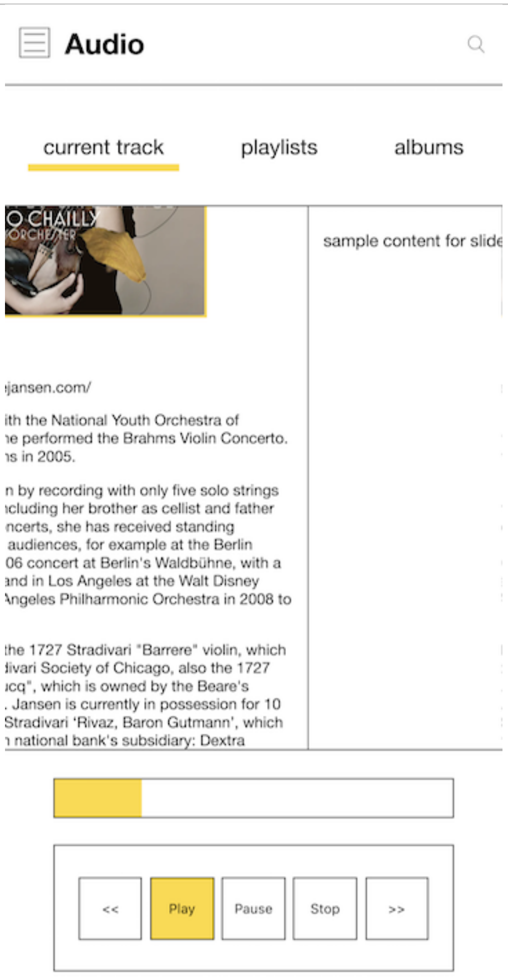
Mobile Design - Touch Events & Interaction

Image - Mobile Design - Touch Events & Interaction - Basic Audio


Audio
current track
playlists
albums
Janine Jansen
Mendelssohn/Bruch: Violin Concertos
Bruch: Violin Concerto No.1 in G minor, Op.26 - 3. Finale (Allegro energico)
<<
Play
Pause
Stop
>>

Mobile Design - Touch Events & Interaction - Basic Audio

Image - Mobile Design - Touch Events & Interaction - Basic Audio - Scroll



sample content for slide

ijansen.com/

ith the National Youth Orchestra of
ie performed the Brahms Violin Concerto.
is in 2005.

n by recording with only five solo strings
icluding her brother as cellist and father
ncerts, she has received standing
audiences, for example at the Berlin
06 concert at Berlin's Waldbühne, with a
and in Los Angeles at the Walt Disney
Angeles Philharmonic Orchestra in 2008 to

the 1727 Stradivari "Barrere" violin, which
livari Society of Chicago, also the 1727
icq", which is owned by the Beare's
. Jansen is currently in possession for 10
Stradivari 'Rivaz, Baron Gutmann', which
i national bank's subsidiary: Dextra

Mobile Design - Touch Events & Interaction - Basic Audio - Scroll

Cordova app - templates - basic

- Cordova default template for project structure
  - *`create` command used for basic structure...*
- create custom, reusable template for a new project
  - *e.g. create starting template for tabs, menu &c. based app...*
- to create a custom template
  - *start with new project structure for Cordova*
  - *then modify to create and configure app structure*
  - *set required icons, splashscreens, designs &c. for template*
- then we can start to package a reusable template

Cordova app - templates - structure

- each template uses the following directory structure

```
|-- template_package
    |__ package.json
    |__ index.js
    |__ template_src
        |__ ... (app template contents...)
```

- template specific code is added to `template_src` directory
- `package.json` includes reference to template's `index.js` file
- `index.js` used to export reference to `template_src` directory

Cordova app - templates - `template_src`

- `template_src` usually includes the following structure

```
|-- hooks (add custom hooks for template, app &c...)
|-- www
    |__ css
        |__ index.css
    |__ img
        |__ logo.png
    |__ js
        |__ index.js
    |__ index.html
|-- config.xml
```

- add any custom scripts to the `hooks` directory
- design and build our template in the `www` directory
- `template_src/config.xml` will usually follow pattern of default Cordova config
- then add template customisations, e.g.
  - *name, description, icons, splashscreens...if necessary*

Cordova app - templates - `package.json`

- `package.json` includes template specific metadata
  - *add keyword `cordova:template` & `ecosystem:cordova`*
  - *used for package distribution, e.g. NPM*
- add reference to `index.js`

```
"main": "index.js"
```

- output will be similar to a standard NPM `package.json` file
  - *created for NPM package management*
  - *then initialised using the command,*

```
npm init
```

Cordova app - templates - template `index.js`

- then add necessary export reference for `template_src` to our template `index.js` file
  - *follows a standard pattern*

```
var path = require('path');

module.exports = {
    dirname : path.join(__dirname, 'template_src')
};
```

Cordova app - templates - finish & create

- template is now ready to be published and shared online
  - *use NPM, GitHub, &c.*
- use as the template for a new local project

```
cordova create basic com.example.basic BasicTemplate --template <path-to-template>
```

- add the local directory path for the custom template
  - *replace `<path-to-template` with local directory for template...*
- creates new Cordova project with custom template
  - *uses `template_src` for the project*

Cordova app - API plugin examples

- a few API plugins to consider
  - *accelerometer*
  - *camera*
  - *connection*
  - *device*
  - *file*
  - *geolocation*
  - *InAppBrowser*
  - *media and capture*
  - *notification*
  - *StatusBar*
  - ...

Data considerations in mobile apps

- no one size fits all model for mobile
- can't just default to the server-side for reading and writing data
- our app may become useless if we rely heavily on remote data
  - *lose our network connection*
  - *run out of monthly data allowance*
  - *or end up with throttled or restricted data on a poor network, e.g. 2G*
- Facebook's introduction of **2G Tuesdays**
  - *remind employees, developers of 2G limitations and issues around the world*
- also need to consider
  - *data security, read and write privileges for certain data stores, authentication for remote sources...*
- careful consideration of the options for reading and writing data
  - *a crucial aspect of our app's planning and subsequent development*

Cordova app - API plugin examples - plugin test 3

setup

- create our initial plugin test shell application

```
cordova create plugintest3 com.example.plugintest plugintest3
```

- add any required plaforms, e.g. Android, iOS, Windows Phone...
  - *we'll add iOS as well*

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

- then update the default www directory
- modify the initial settings in our app's `config.xml` file
- then run an initial test to ensure the shell application loads correctly
  - *run in the Android emulator or*
  - *run on a connected Android device*

```
cordova emulate android
```

- or

```
cordova run android
```

Cordova app - API plugin examples - plugin test 3

setup

- also add support for iOS development

```
cordova platform add ios --save
```

- running a test application on iOS is not as simple as Android
- need to add support to Cordova for a local iOS simulator
  - *add package for iOS simulator using* ***npm***
  - ***NB****: may require admin or sudo permissions to install correctly*

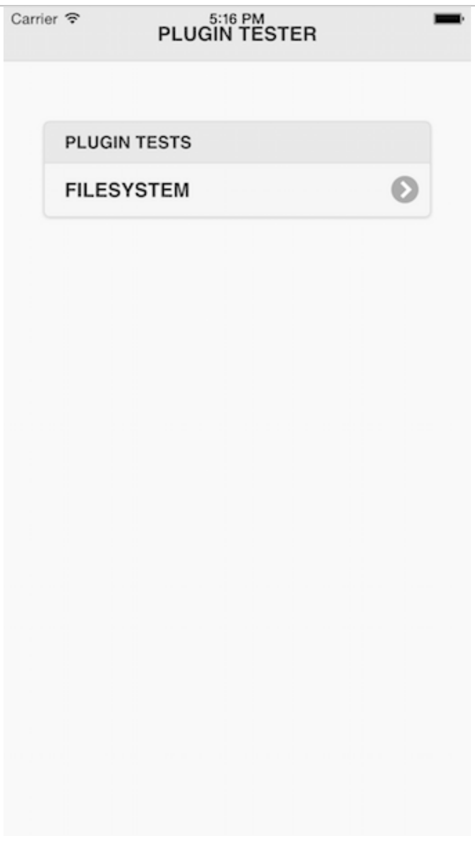
```
npm install -g ios-sim
```

- then run our Cordova app from the working directory

```
cordova run ios
```

- Cordova will try to load the application using this local simulator
  - *without defaulting to Xcode*
- quickly test our iOS application with this simulator

Image - iOS Local Simulator



iOS Simulator - running locally on OS X

Cordova app - API plugin examples - plugin test 3

iOS simulator - options

- iOS simulator gives us many useful options
  - *helpful ways to test our local Cordova based iOS applications*
- emulate many different devices
  - *from the iPhone SE to the iPhone X and various iPads...*
- mimic many of these device's hardware features
  - *such as rotate, shake, different keyboards...*
  - *also output to a simulated Apple Watch device*
- various debugging options available within this simulator
  - *including ability to mimic locations for GPS enabled applications*
- quickly take a screenshot of the current application screen within the simulator

Cordova app - API plugin examples - plugin test 3

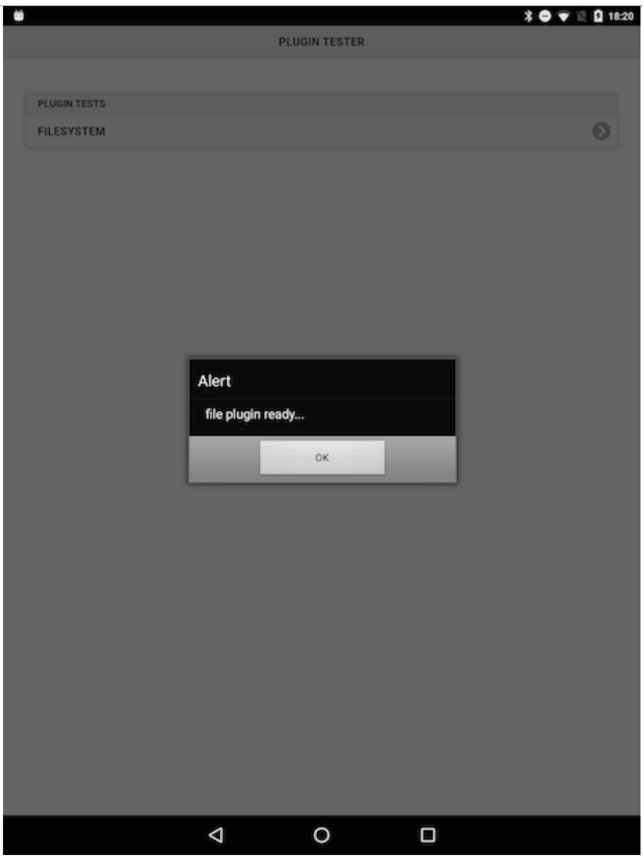
plugins - add filesystem

- add and use the **file** plugin
- plugin has been designed to permit read and write access to files
  - *files are stored on the local device for Cordova applications*
- **file** plugin is initially based on open specifications
  - *includes the **HTML5 File API**, W3C's **FileWriter** specification...*
- add the file plugin to our test application using the standard CLI command

```
cordova plugin add cordova-plugin-file
```

- command will install plugin for all currently installed platforms
  - *includes Android and iOS for our test application*

Image - API Plugin Tester - file



Alert

file plugin ready...

API Plugin Tester - file plugin ready

Cordova app - API plugin examples - plugin test 3

plugins - test filesystem

- using this plugin we can read local files from within the filesystem
- we could read a file from within our Cordova application
  - *e.g. located in the following directory*

```
...
|- www
   |- docs
      |- txt
         |- madeira.txt
```

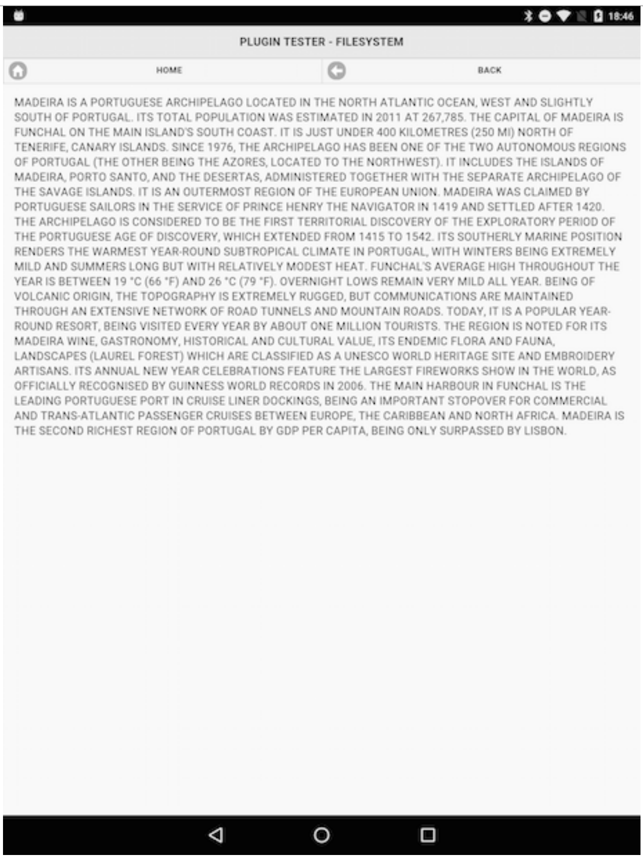
- we can use the available global `cordova.file` object
- to be able to use the URL for our text document in the file-system directory
  - *convert it to a `DirectoryEntry` using*

```
window.resolveLocalFileSystemURL()
```

- in our standard `onDeviceReady()` function
  - *use this global object to resolve the URL of our file*
  - *then pass to specified callbacks for success and fail*

```
window.resolveLocalFileSystemURL(cordova.file.applicationDirectory +
"www/docs/txt/madeira.txt", onSuccess, onFail);
```

Image - API Plugin Tester - file



MADEIRA IS A PORTUGUESE ARCHIPELAGO LOCATED IN THE NORTH ATLANTIC OCEAN, WEST AND SLIGHTLY SOUTH OF PORTUGAL. ITS TOTAL POPULATION WAS ESTIMATED IN 2011 AT 267,785. THE CAPITAL OF MADEIRA IS FUNCHAL ON THE MAIN ISLAND'S SOUTH COAST. IT IS JUST UNDER 400 KILOMETRES (250 MI) NORTH OF TENERIFE, CANARY ISLANDS. SINCE 1976, THE ARCHIPELAGO HAS BEEN ONE OF THE TWO AUTONOMOUS REGIONS OF PORTUGAL (THE OTHER BEING THE AZORES, LOCATED TO THE NORTHWEST). IT INCLUDES THE ISLANDS OF MADEIRA, PORTO SANTO, AND THE DESERTAS, ADMINISTERED TOGETHER WITH THE SEPARATE ARCHIPELAGO OF THE SAVAGE ISLANDS. IT IS AN OUTERMOST REGION OF THE EUROPEAN UNION. MADEIRA WAS CLAIMED BY PORTUGUESE SAILORS IN THE SERVICE OF PRINCE HENRY THE NAVIGATOR IN 1419 AND SETTLED AFTER 1420. THE ARCHIPELAGO IS CONSIDERED TO BE THE FIRST TERRITORIAL DISCOVERY OF THE EXPLORATORY PERIOD OF THE PORTUGUESE AGE OF DISCOVERY, WHICH EXTENDED FROM 1415 TO 1542. ITS SOUTHERLY MARINE POSITION RENDERS THE WARMEST YEAR-ROUND SUBTROPICAL CLIMATE IN PORTUGAL, WITH WINTERS BEING EXTREMELY MILD AND SUMMERS LONG BUT WITH RELATIVELY MODEST HEAT. FUNCHAL'S AVERAGE HIGH THROUGHOUT THE YEAR IS BETWEEN 19 °C (66 °F) AND 26 °C (79 °F). OVERNIGHT LOWS REMAIN VERY MILD ALL YEAR. BEING OF VOLCANIC ORIGIN, THE TOPOGRAPHY IS EXTREMELY RUGGED, BUT COMMUNICATIONS ARE MAINTAINED THROUGH AN EXTENSIVE NETWORK OF ROAD TUNNELS AND MOUNTAIN ROADS. TODAY, IT IS A POPULAR YEAR-ROUND RESORT, BEING VISITED EVERY YEAR BY ABOUT ONE MILLION TOURISTS. THE REGION IS NOTED FOR ITS MADEIRA WINE, GASTRONOMY, HISTORICAL AND CULTURAL VALUE, ITS ENDEMIC FLORA AND FAUNA, LANDSCAPES (LAUREL FOREST) WHICH ARE CLASSIFIED AS A UNESCO WORLD HERITAGE SITE AND EMBROIDERY ARTISANS. ITS ANNUAL NEW YEAR CELEBRATIONS FEATURE THE LARGEST FIREWORKS SHOW IN THE WORLD, AS OFFICIALLY RECOGNISED BY GUINNESS WORLD RECORDS IN 2006. THE MAIN HARBOUR IN FUNCHAL IS THE LEADING PORTUGUESE PORT IN CRUISE LINER DOCKINGS, BEING AN IMPORTANT STOPOVER FOR COMMERCIAL AND TRANS-ATLANTIC PASSENGER CRUISES BETWEEN EUROPE, THE CARIBBEAN AND NORTH AFRICA. MADEIRA IS THE SECOND RICHEST REGION OF PORTUGAL BY GDP PER CAPITA, BEING ONLY SURPASSED BY LISBON.

API Plugin Tester - read an app txt file

Cordova app - API plugin examples - plugin test 3

plugins - test filesystem `onSuccess`

- render this text after retrieving from the requested file
  - *update our `onSuccess()` function to output the file's content*

```
function onSuccess(data) {
  data.file(function(file) {
    var readFile = new FileReader();
    readFile.onloadend = function(e) {
      //output result as required by app...
      // e.g this.result
    }
    readFile.readAsText(file);
  });
}
```

- call the `file()` method on our returned file data
  - *effectively gives us a hook/handle into the file*
  - *we can now work with the returned file data*
- then call the `FileReader()` method from the **FileAPI**
  - *and process the returned text*
- output to our specified HTML element
  - *using a standard selector with the `html()` method*

Cordova app - API plugin examples - plugin test 3

plugins - test filesystem `onFail()`

- complement to the `onSuccess()` function
- now add our function `onFail()` for the fail callback
- test it with the returned error code

```
function onFail(error) {
  console.log("FileSystem Error"+error.code);
  // output error and code as required in app...
  // e.g error.code
}
```

- uses the passed error object
  - *returns a code for rendering in the specified selector*
- obviously does not make a lot of sense to our user

Cordova app - API plugin examples - plugin test 3

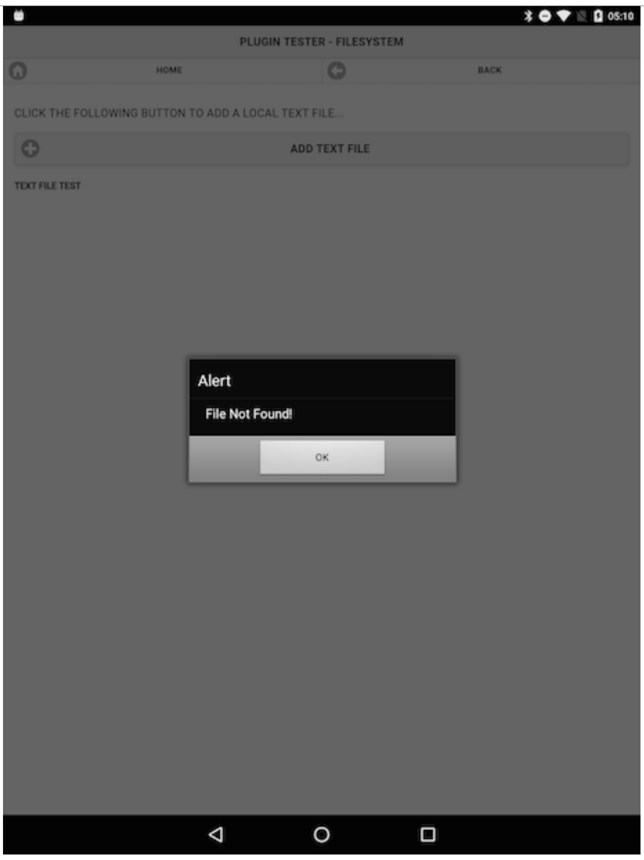
plugins - test filesystem `onFail()`

- we can use a conditional statement to check for certain returned error codes
  - *then output a meaningful error message to the user in the application*

```
function onFail(error) {
  switch(error.code) {
    case 1:
    alert('File Not Found!');
    break;
    //add other options to cover additional error codes...
    default:
    alert('An error occurred reading this file.');
  };
}
```

- now output more graceful error messages and feedback to the user

Image - API Plugin Tester - file


05:10
PLUGIN TESTER - FILESYSTEM
HOME
BACK
CLICK THE FOLLOWING BUTTON TO ADD A LOCAL TEXT FILE...
ADD TEXT FILE
TEXT FILE TEST
Alert
File Not Found!
OK

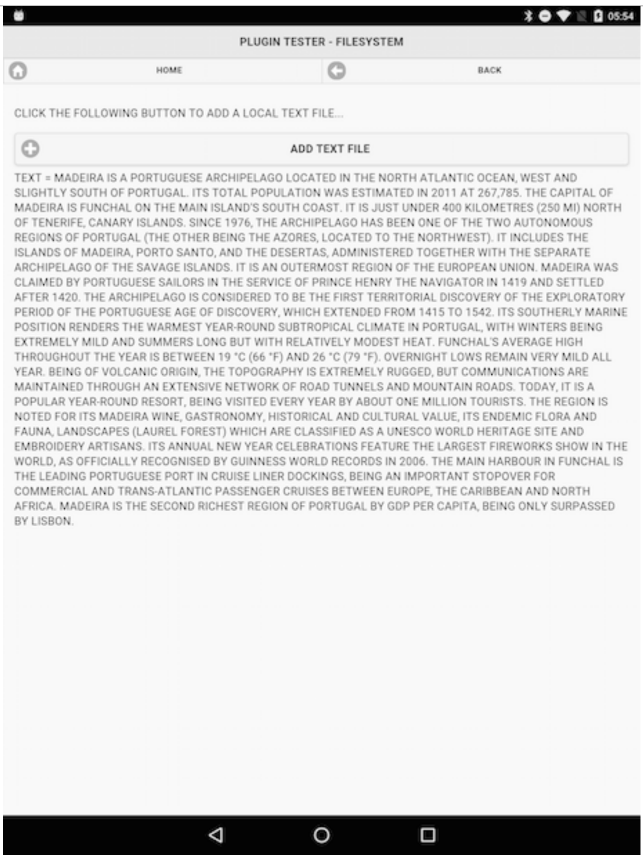
API Plugin Tester - output error message

Cordova app - API plugin examples - plugin test 3

plugins - test filesystem with event

- easily link file loading to a given event, such as a user tap event
- instead of loading the file by default with the `onDeviceReady()` function
  - *get the contents of our file when needed by the user*
- link this to a button event, a separate page init event...
  - *touch event on button, link &c.*
- then call our local file as before within its own function, `getTxtFile()`

Image - API Plugin Tester - file



CLICK THE FOLLOWING BUTTON TO ADD A LOCAL TEXT FILE...

TEXT = MADEIRA IS A PORTUGUESE ARCHIPELAGO LOCATED IN THE NORTH ATLANTIC OCEAN, WEST AND SLIGHTLY SOUTH OF PORTUGAL. ITS TOTAL POPULATION WAS ESTIMATED IN 2011 AT 267,785. THE CAPITAL OF MADEIRA IS FUNCHAL ON THE MAIN ISLAND'S SOUTH COAST. IT IS JUST UNDER 400 KILOMETRES (250 MI) NORTH OF TENERIFE, CANARY ISLANDS. SINCE 1976, THE ARCHIPELAGO HAS BEEN ONE OF THE TWO AUTONOMOUS REGIONS OF PORTUGAL (THE OTHER BEING THE AZORES, LOCATED TO THE NORTHWEST). IT INCLUDES THE ISLANDS OF MADEIRA, PORTO SANTO, AND THE DESERTAS, ADMINISTERED TOGETHER WITH THE SEPARATE ARCHIPELAGO OF THE SAVAGE ISLANDS. IT IS AN OUTERMOST REGION OF THE EUROPEAN UNION. MADEIRA WAS CLAIMED BY PORTUGUESE SAILORS IN THE SERVICE OF PRINCE HENRY THE NAVIGATOR IN 1419 AND SETTLED AFTER 1420. THE ARCHIPELAGO IS CONSIDERED TO BE THE FIRST TERRITORIAL DISCOVERY OF THE EXPLORATORY PERIOD OF THE PORTUGUESE AGE OF DISCOVERY, WHICH EXTENDED FROM 1415 TO 1542. ITS SOUTHERLY MARINE POSITION RENDERS THE WARMEST YEAR-ROUND SUBTROPICAL CLIMATE IN PORTUGAL, WITH WINTERS BEING EXTREMELY MILD AND SUMMERS LONG BUT WITH RELATIVELY MODEST HEAT. FUNCHAL'S AVERAGE HIGH THROUGHOUT THE YEAR IS BETWEEN 19 °C (66 °F) AND 26 °C (79 °F). OVERNIGHT LOWS REMAIN VERY MILD ALL YEAR. BEING OF VOLCANIC ORIGIN, THE TOPOGRAPHY IS EXTREMELY RUGGED, BUT COMMUNICATIONS ARE MAINTAINED THROUGH AN EXTENSIVE NETWORK OF ROAD TUNNELS AND MOUNTAIN ROADS. TODAY, IT IS A POPULAR YEAR-ROUND RESORT, BEING VISITED EVERY YEAR BY ABOUT ONE MILLION TOURISTS. THE REGION IS NOTED FOR ITS MADEIRA WINE, GASTRONOMY, HISTORICAL AND CULTURAL VALUE, ITS ENDEMIC FLORA AND FAUNA, LANDSCAPES (LAUREL FOREST) WHICH ARE CLASSIFIED AS A UNESCO WORLD HERITAGE SITE AND EMBROIDERY ARTISANS. ITS ANNUAL NEW YEAR CELEBRATIONS FEATURE THE LARGEST FIREWORKS SHOW IN THE WORLD, AS OFFICIALLY RECOGNISED BY GUINNESS WORLD RECORDS IN 2006. THE MAIN HARBOUR IN FUNCHAL IS THE LEADING PORTUGUESE PORT IN CRUISE LINER DOCKINGS, BEING AN IMPORTANT STOPOVER FOR COMMERCIAL AND TRANS-ATLANTIC PASSENGER CRUISES BETWEEN EUROPE, THE CARIBBEAN AND NORTH AFRICA. MADEIRA IS THE SECOND RICHEST REGION OF PORTUGAL BY GDP PER CAPITA, BEING ONLY SURPASSED BY LISBON.

API Plugin Tester - event file loader

Cordova app - API plugin examples - plugin test 3

plugins - test filesystem with file write

- now read files from the local device's native storage thanks to Cordova's File plugin
- file plugin also offers an option to write to files in the same local filesystem
- quickly create a test app for writing to files
- create your project
- `cd` to app's working directory
- add required platforms
- add our required Cordova API plugin for working with the file system
- run usual initial tests for app loading, `deviceready` event...

Cordova app - API plugin examples - plugin test 3

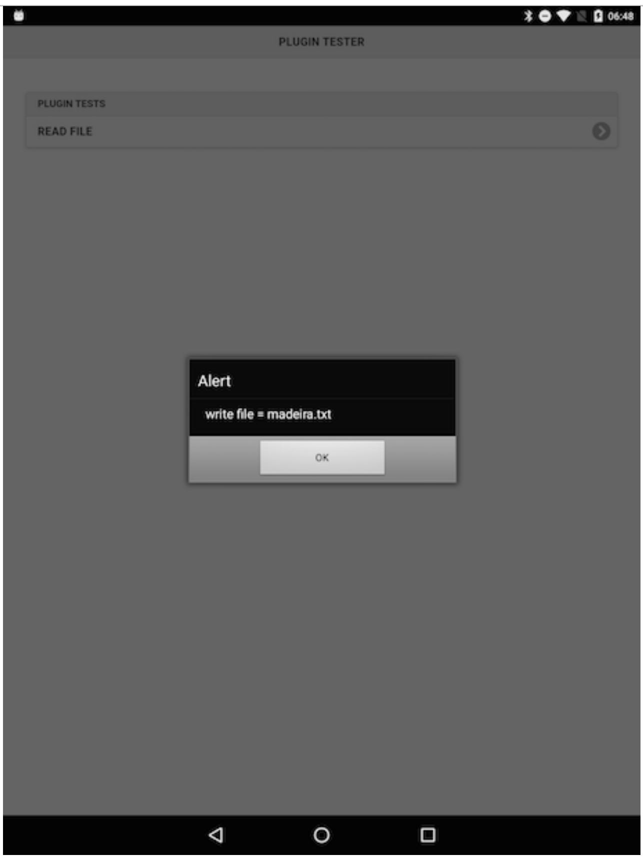
plugins - test filesystem with file write

- now start to add writing to a file to our test app
- start, as we did with file reading, by getting a hook/handle to a file
- we can then write to a file within the assigned app's data directory
  - *specific app directory has read and write access*
  - *allows us to create files as needed for our app*
  - *then read and write within the confines of the native app*
- use `window.resolveLocalFileSystemURL` to allow us to work with this data directory

```
var fileDir = cordova.file.dataDirectory;
window.resolveLocalFileSystemURL(fileDir, function(dir) {
// do something useful...
});
```

- in application specific directory get our required file for writing

Image - API Plugin Tester - file



write file = madeira.txt

API Plugin Tester - get file for writing

Cordova app - API plugin examples - plugin test 3

plugins - test filesystem with file write

- create a new file if it doesn't exist on app loading
- use directory object with `getFile()` method etc...
  - *set flag to create a new file*

```
window.resolveLocalFileSystemURL(fileDir, function(dir) {
dir.getFile("madeira.txt", {create:true}, function(file) {
//do something useful
});
});
```

- pass file object to other functions for processing...
- create our write function to check and write to specified file within app's data directory

Cordova app - API plugin examples - plugin test 3

plugins - test filesystem with file write

- now write some simple text to our file

```
function writeTxtFile(data) {
  //check passed data for writing
  if (data !== "") {
    //new text to write to file
    var text = data;
    //use write file object
    writeObj.createWriter(function(writeFile) {
      //call seek() to ensure we append to end of file
      writeFile.seek(writeFile.length);
      //create raw Blob for writing
      var textBlob = new Blob([text], {type:'text/plain'});
      //write to the end of the file
      writeFile.write(textBlob);
    });
  }
}
```

Cordova app - API plugin examples - plugin test 3

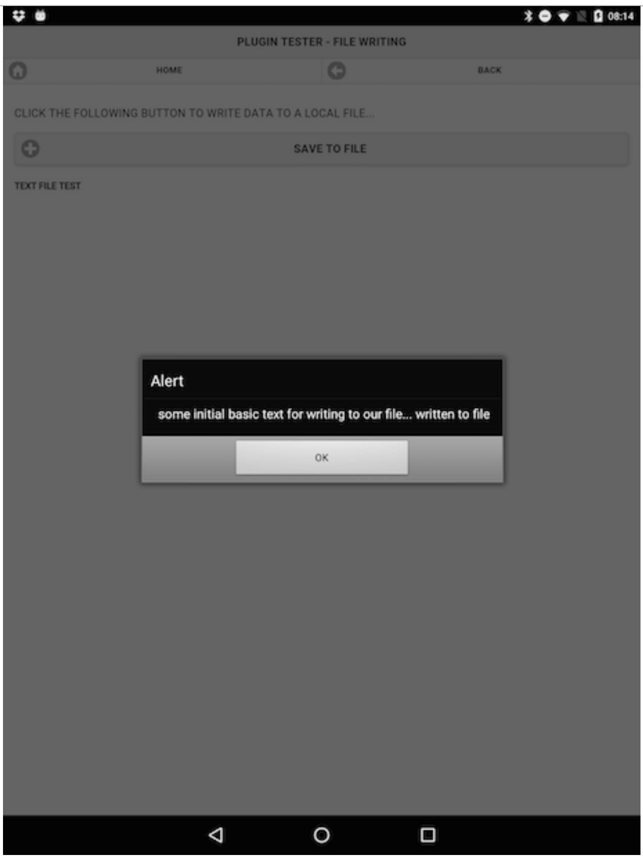
plugins - test filesystem with file write

- then call this `writeTxt()` as needed within our application
  - *e.g. calling it from event handler for a button tap*
- could easily get text to write from an input field, from metadata...
- then pass it to our `writeTxtFile()` function for writing

e.g.

```
writeTxtFile("some initial basic text for writing to our file...");
```

Image - API Plugin Tester - file

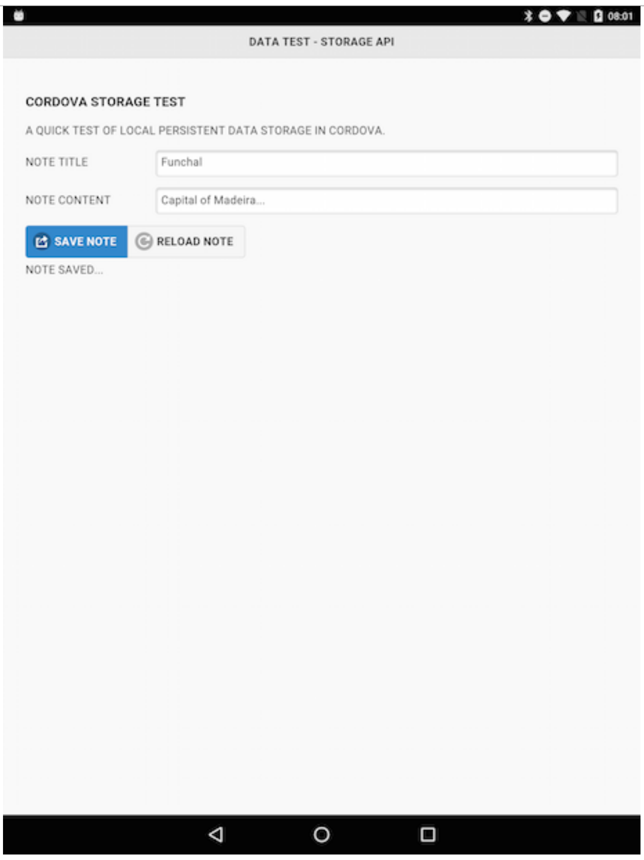

08:14
PLUGIN TESTER - FILE WRITING
HOME
BACK
CLICK THE FOLLOWING BUTTON TO WRITE DATA TO A LOCAL FILE...
SAVE TO FILE
TEXT FILE TEST
OK

Alert

some initial basic text for writing to our file... written to file

API Plugin Tester - text written to file

Image - Data Tester



CORDOVA STORAGE TEST

A QUICK TEST OF LOCAL PERSISTENT DATA STORAGE IN CORDOVA.

NOTE TITLE Funchal

NOTE CONTENT Capital of Madeira...

NOTE SAVED...

DataTest1 - save a note

Cordova app - LocalStorage - data test 1

app logic - save.js

- need to handle events for our `reloadNote` button
- retrieve our notes data
  - *loaded by calling the `reloadNoteData()` function*
- uses the main app object, `storageNotes`
  - *gets the defined key for our notes*
- use this key to retrieve stored *stringified* JSON object
- then use `JSON.parse()` to convert the *stringified* object to a plain JSON object
  - *contains our note information*
- use this note information
  - *populate form fields*
  - *output our notes for rendering to the DOM*

Cordova app - LocalStorage - data test 1

app logic - save.js - reload button handler

- event handler for reload button
  - *call `reloadNoteData()`*
  - *output and update result...*
- reload note data

```
function reloadNoteData() {
  var noteInfo = JSON.parse(storageNotes.get(NOTE_KEY));
  loadFormFields(noteInfo);
  noteOutput(noteInfo);
}
```

- load form fields data

```
function loadFormFields(data) {
  if (data) {
    document.getElementById('noteName').value = data.noteName;
    document.getElementById('noteContent').value = data.noteContent;
  }
}
```

Cordova app - LocalStorage - data test 1

app logic - save.js

- pageinit event
  - *eg: check and validate the rendered form for our notes*
- to validate our form we specify
  - *a set of options as a parameter to* `validate()`
  - *many different options available*
  - *eg: add a* `rules` *object,* `messages` *object...*
- in the `rules` object
  - *set both input fields as required*
- then reload our note data
  - *update the application accordingly*

Cordova app - LocalStorage - data test 1

app logic - save.js - `pageshow` event

```
$("#noteForm").validate({
  rules: {
    noteName: "required",
    noteContent: "required"
  },
  messages: {
    noteName: "Add title for note",
    noteContent: "Add your note"
  }
});
```

Cordova app - LocalStorage - data test 1

app logic - storagenotes.js

- add another new JS file, `storagenotes.js`
  - *store the logic for getting and setting of data with `localStorage`*
- start by creating a singleton object for this instance
- creating this object to ensure that we only have one instance
- create this object by calling the `getInstance()` function
  - *in effect, the guardian to the instance object for the application*
- function also highlights a pattern known as `Lazy Load`
  - *checks to see if an instance has already been created*
- if not, create one and then store for future reference
- all subsequent calls will now received this stored reference
- this pattern is particularly useful for mobile development
- helps us save CPU and memory usage within an application
  - *an object is only created when it is actually needed*
- gives us a single object with getters and setters for the local storage

Cordova app - LocalStorage - data test 1

app logic - storagenotes.js

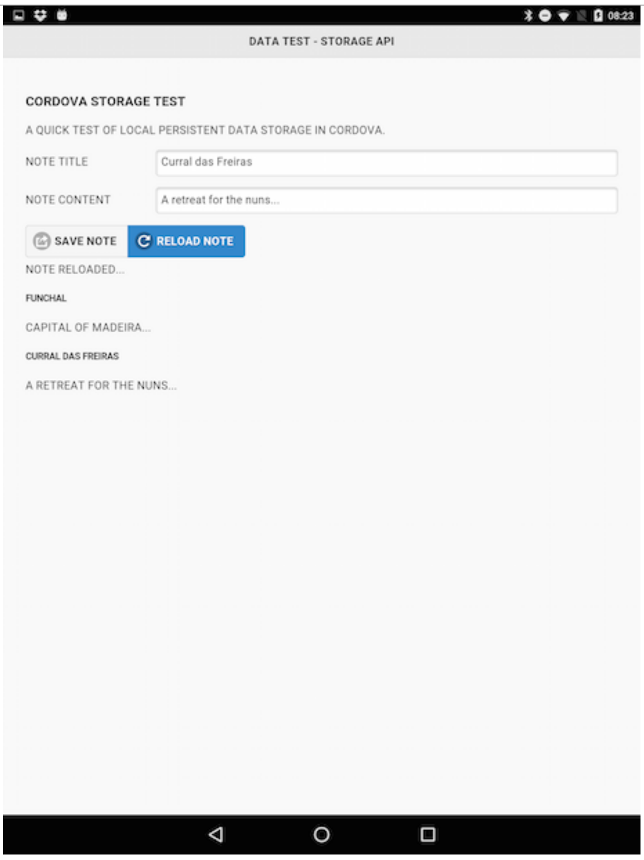
```
var NotesManager = (function () {
  var instance;

  function createNoteObject() {
      return {
        set: function (key, value) {
          window.localStorage.setItem(key, value);
        },
        get: function (key) {
          return window.localStorage.getItem(key);
        }
      };
  };

  return {
    getInstance: function () {
      if (!instance) {
        instance = createNoteObject();
      }
      return instance;
    }
  };

})();
```

Image - Data Tester


DATA TEST - STORAGE API

CORDOVA STORAGE TEST

A QUICK TEST OF LOCAL PERSISTENT DATA STORAGE IN CORDOVA.

NOTE TITLE Curral das Freiras

NOTE CONTENT A retreat for the nuns...

NOTE RELOADED...

FUNCHAL

CAPITAL OF MADEIRA...

CURRAL DAS FREIRAS

A RETREAT FOR THE NUNS...

DataTest1 - update the notes

References

- Cordova Docs - Events
- Cordova API
  - *config.xml*
  - *plugins*
  - *plugin - device*
  - *plugin - file*
  - *plugin - media*
  - *plugin - Splashscreen*
- HTML5
  - *HTML5 File API*