Cordova - Guide - App Development - Basics

- Dr Nick Hayward

A brief overview and introduction to Apache Cordova application development and design.

Contents

Cordova CLI - build initial project

Cordova app development begins with creation of a new app using the Cordova CLI tool.

An example pattern and usage is as follows for initial project creation.

```
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 App - structure recap - app directory

A newly created project will include the following type of structure for design and development.

e.g.

```
|- 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

`www` directory

Then, an example `www` directory will be as follows,

e.g.

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

`index.html`

The initial `index.html` file will be as follows.

```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>
```

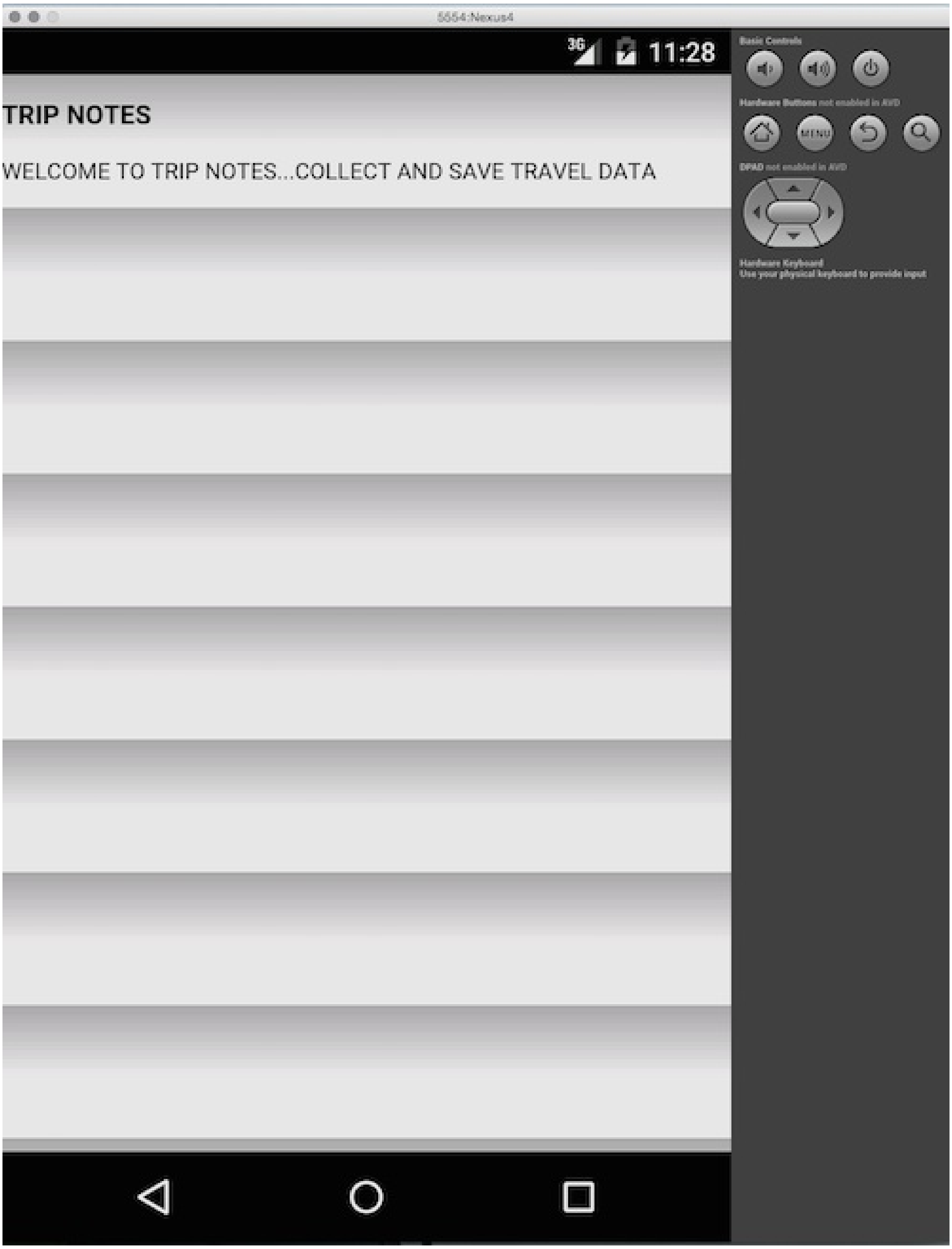
The first thing we'll do is clear out the body of our HTML document, and then start adding some of our own content and structure. The `cordova.js` script reference is always added to an app, but it is not something we need to explicitly add to the app install itself. It is provided, by default, as part of the Cordova create and build for a current project.

An example updated file might be as follows,

```
<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>
```

A lack of styling will be an issue.

Image - Cordova App - Basic v0.01


5554:Nexus4
11:28
Basic Controls
Hardware Buttons not enabled in AVD
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input

TRIP NOTES

WELCOME TO TRIP NOTES...COLLECT AND SAVE TRAVEL DATA

add Cordova specifics

By default, the Cordova container for the application will expose native APIs to our web application running in the WebView. However, in general these APIs will not be available until an applicable plugin has been added to the project. The container also needs to perform some preparation before the APIs can be used.

To help us as developers, Cordova informs us when the container, and associated APIs, are ready for use. It fires a specific event, called the `deviceready` event. In effect, any logic within our application that requires use of Cordova

APIs should be executed after this `deviceready` notification has been received.

add some jQuery

- add to foot of `<body>`

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

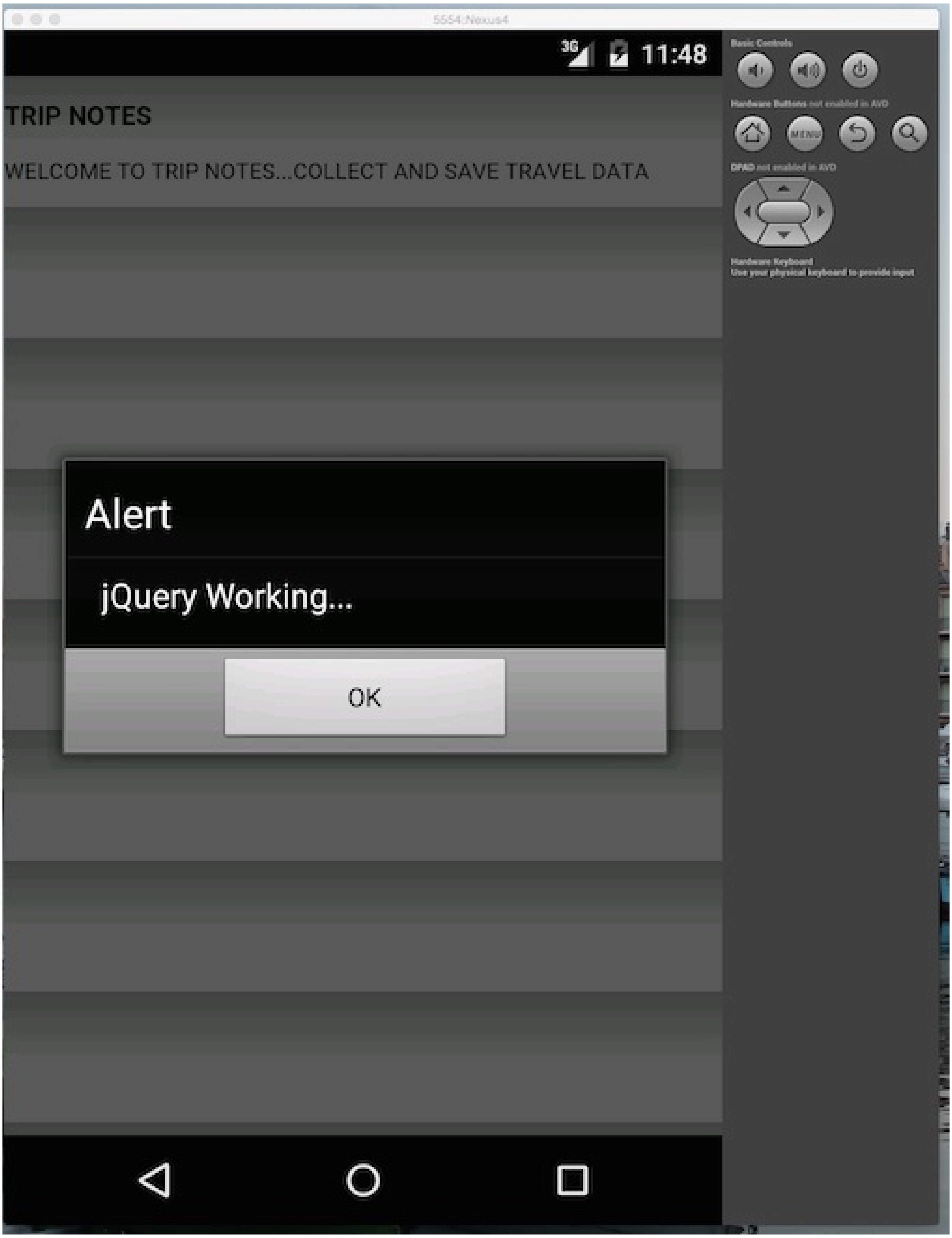
- add test to `trip.js` file

```
function tripNotes() {
  alert("JS Working...");
}

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

We can also add some jQuery, or another required JS library or plain JS, and then use the app's container with standard JS functions. For example, outputting a simple alert.

Image - Cordova App - Basic v0.02


5554:Nexus4
3G
11:48
Basic Controls
Hardware Buttons not enabled in AVD
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input

TRIP NOTES

WELCOME TO TRIP NOTES...COLLECT AND SAVE TRAVEL DATA

Alert

jQuery Working...

OK

add some jQuery Mobile

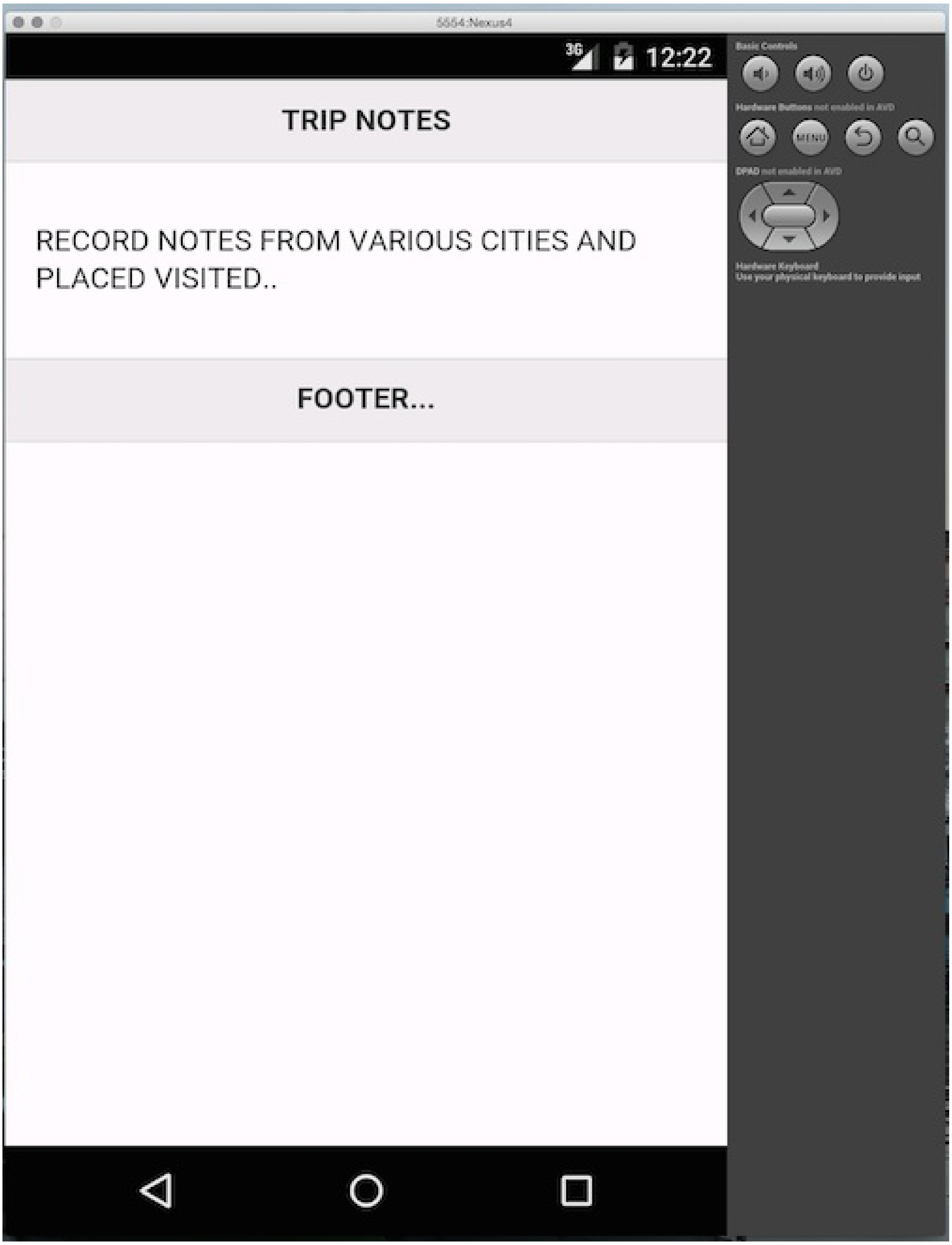
- update `head` with local jQuery Mobile CSS

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

- update `body` for basic app

```html
<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


5554:Nexus4
3G
12:22
Basic Controls
Hardware Buttons not enabled in AVD
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input

TRIP NOTES

RECORD NOTES FROM VARIOUS CITIES AND PLACED VISITED..

FOOTER...

jQuery Mobile - test transitions

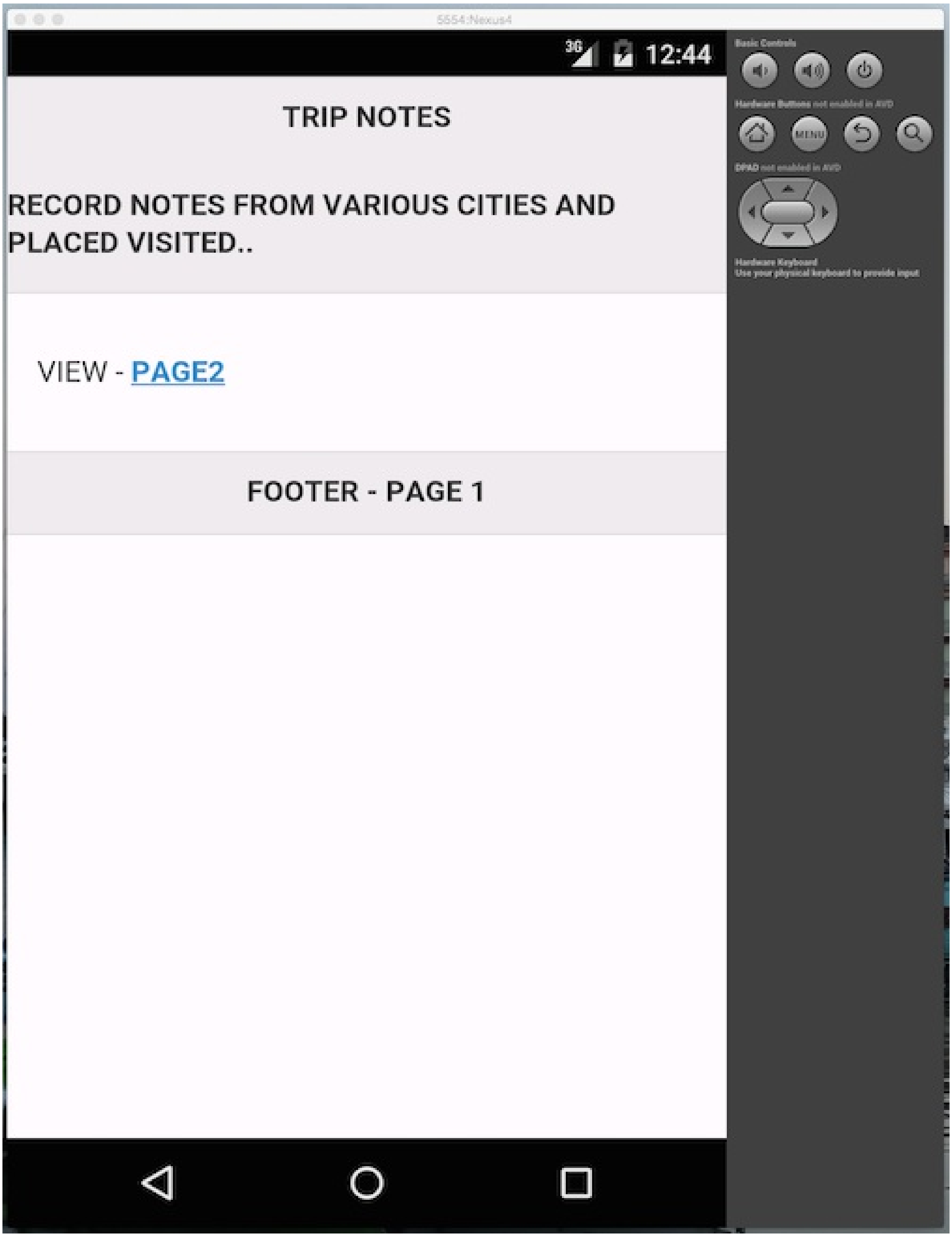
We can also add some jQuery Mobile transitions, and test some basic internal navigation.

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

```
<!-- page1 -->
<div data-role="page" id="page1">
```

```html
  <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


5554:Nexus4
12:44
Basic Controls
Hardware Buttons not enabled in AVD
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input

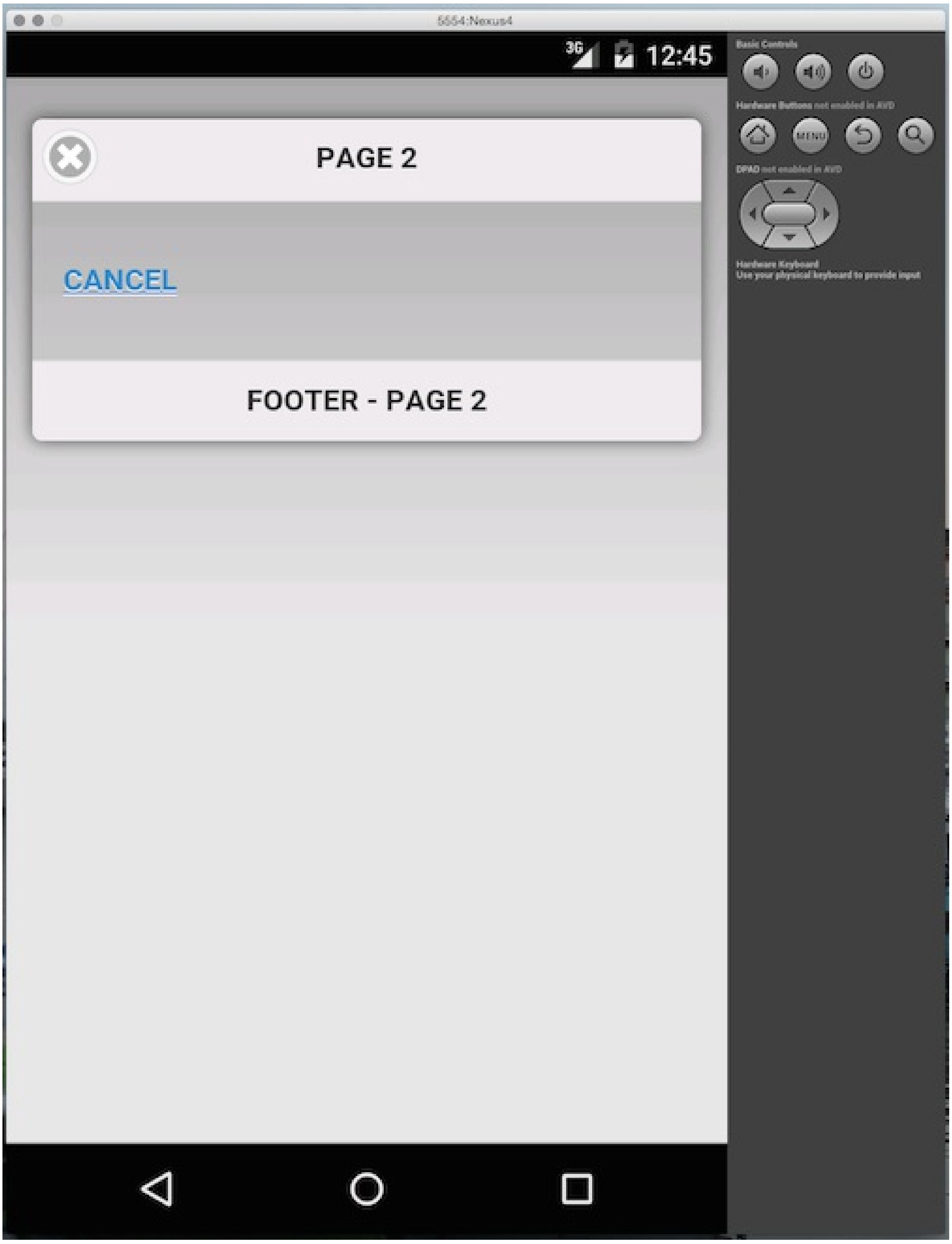
TRIP NOTES

RECORD NOTES FROM VARIOUS CITIES AND PLACED VISITED..

VIEW - PAGE2

FOOTER - PAGE 1

Image - Cordova App - Basic v0.05



jQuery Mobile - navigation

We may also use jQuery Mobile to manage the navigation stack within the Cordova app.

intro

So, we've just added some jQuery Mobile to an initial Cordova app. This included some initial pages, transitions, and basic navigation.

To help us build this out with jQuery Mobile, for example, we'll briefly need to look at navigation within our apps. For the purposes of mobile development, this navigation thankfully follows an **asynchronous** pattern.

So, navigation in jQuery mobile, for example, is based upon loading pages into the DOM using AJAX. This will modify the page's content, and then re-render for display to the user. It will also include a set of aesthetically pleasing, and useful, animations to help inform the user of changes in state, and therefore appropriate updates in the content.

This navigation system effectively hijacks a link within a page's content container, and then routes it through an AJAX request. The benefit for developers is a particularly useful, and almost painless, approach to asynchronous navigation. Most of the time, we are not even aware of this updated request. In spite of hijacking the link request, it is still able to support standard concepts such as **anchors** and use of the **back** button without breaking the coherence and logic of the application.

Therefore, jQuery Mobile is able to load and view groups of disparate content in pages within our initial home document. In essence, the **many** combining to form the **one** coherent application.

Its support for core JavaScript event handling, in particular for URL fragment identifiers with `hashchange` and `popstate`, allows the application to persist, at least temporarily, a record of user navigation and paths through the content. We can also tap into this internal history of the application, and again hijack certain patterns to help us better inform the user about state changes, different paths, content, and so on.

example navigation

The following is an example of using the jQuery Mobile standard navigate method, `$.mobile.navigate`, which is used as a convenient way to track history and navigation events.

With this simple example, we can set our record information for the link, effectively any useful information for the link or affected change in state. We can then log the available direction for navigation, in this example the fact we can go **back**, the url for the nav state, and any available hash. In our example, the simple appended hash in the url, `#nav1`.

What this allows us to do is keep a clear record of user traversal through the application, log it as required by given state changes, and then use it to inform our application's logic, our user, and so on.

- Demo - jQuery Mobile nav

jQuery Mobile - using widgets

Within our app's webview container, we can add standard HTML elements for any required content containers.

For example, standard HTML and HTML5 elements such as `p`, `headings`, `lists`, `sections`, and so on.

Thankfully, we don't necessarily have to build the whole application from scratch.

jQuery Mobile includes a wide-range of pre-fabricated widgets we can add to our applications. These are naturally touch-friendly, and include collapsible elements, forms, responsive tables, dialogs, and many more. We've already seen an example with the overall `pageContainer` widget.

listviews

A good example of this type of pre-fabricated widget is a **listview**.

jQuery Mobile helps us style, render and then manipulate standard data output and collections, including rendering of lists as interactive, animated views.

These lists are coded with a `data-role` attribute, as we saw earlier with a page value for a data role...

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

This allows us to style and render our lists with additional options such as a dynamic search filter.

- Demo - jQuery Mobile listview 1
- Demo - jQuery Mobile listview 2

listviews - example

```html
<!-- 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

```html
<!-- 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

Listviews are a lot of fun and very useful for easily organising our data, as we've just seen.

However, we can also use a **listview** to add filtering and live search options to our lists.

We set a simple client-side filter by adding an attribute for `data-filter`, and then set the value to `true`

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

jQuery Mobile will then add a search query input field to the top of our list widget, and set a filter for the entered search query. Effectively, it's performing a pattern match using a partially entered string against strings in a designated list. It can match partial fragments of a list item, and then dynamically filter our list content.

We can also set some default, helpful text for the input field. Our way of prompting the user to interact with, and therefore use this feature correctly.

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

To tidy up the presentation of our list, we can also add an inset using the attribute

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

Now, we have a much better design for our lists, and some useful filtering options as well.

- Demo - jQuery Mobile listview 4

listviews - adding some formatted content

One of the fun aspects of working with a framework such as jQuery Mobile, and others such as the excellent Ionic framework and OnsenUI, is the simple way we can organise and format our data presentations and views.

For example, if we have a grouped dataset, we can still present it using lists. However, we can also add informative headings, links to different categories within this dataset, and simple styling to help differentiate components within the list interface.

Therefore, we structure the list as normal, with sub-headings, paragraphs, and so on. Then, jQuery Mobile gives us a simple option for setting certain list content as an aside. For example,

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

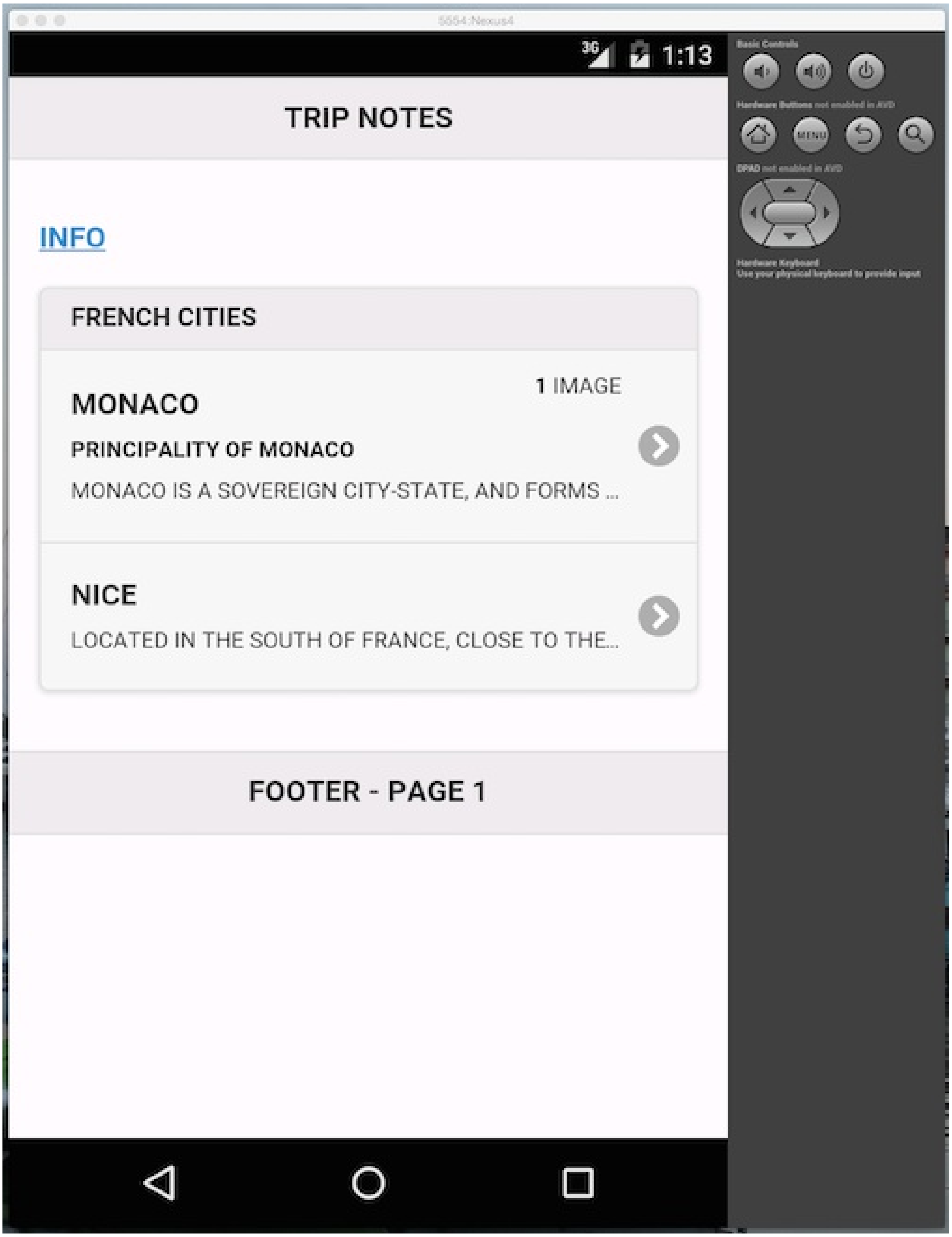
There are many similar tweaks and additions we can add to help improve organisation and rendering of list data. Further details can, of course, be found in the jQuery Mobile API.

listviews - updated 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

Image - jQuery Mobile - add some organisation



5554:Nexus4

1:13

TRIP NOTES

INFO

FRENCH CITIES

MONACO

1 IMAGE

PRINCIPALITY OF MONACO

MONACO IS A SOVEREIGN CITY-STATE, AND FORMS ...

NICE

LOCATED IN THE SOUTH OF FRANCE, CLOSE TO THE...

FOOTER - PAGE 1

Basic Controls
Hardware Buttons not enabled in AVD
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input

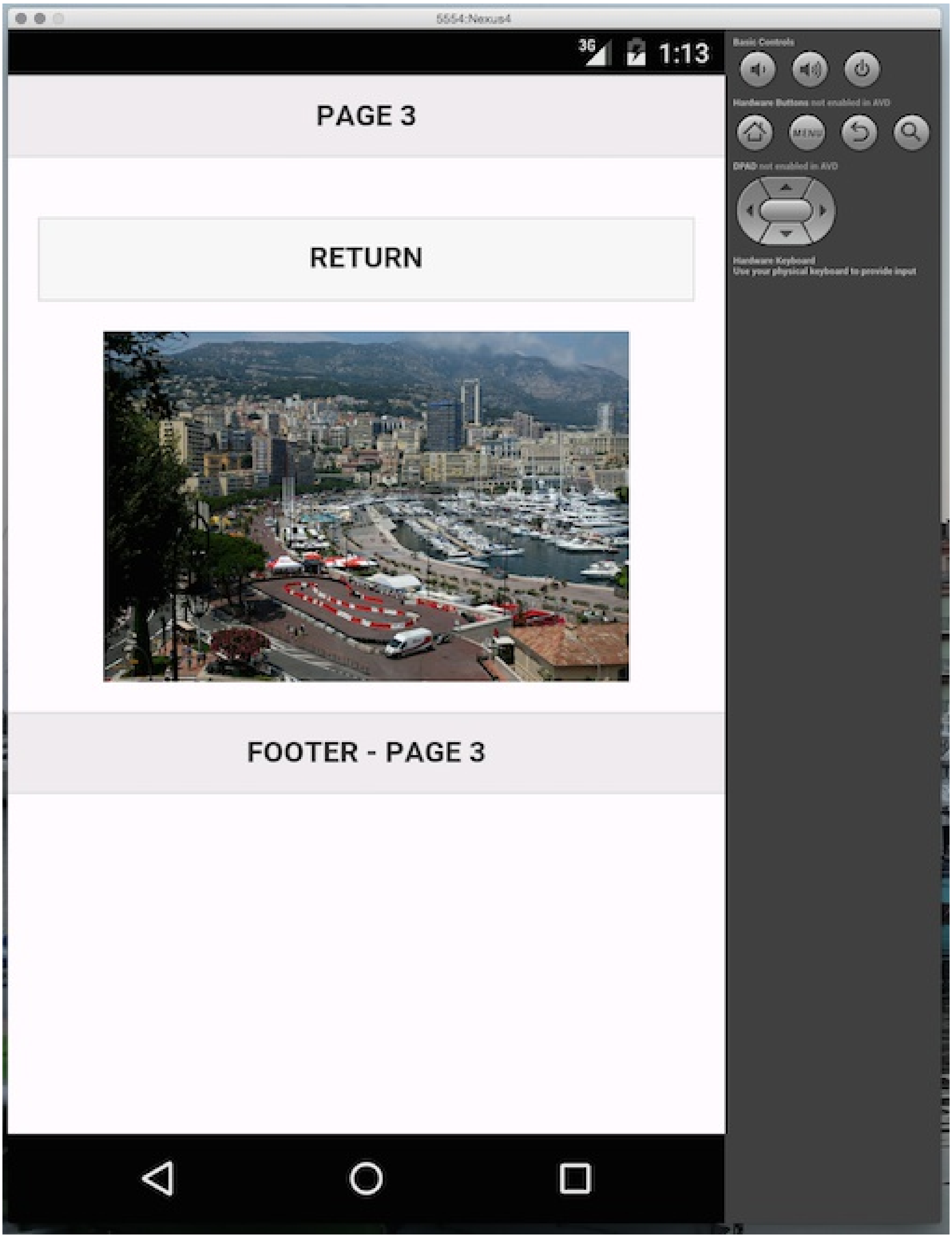
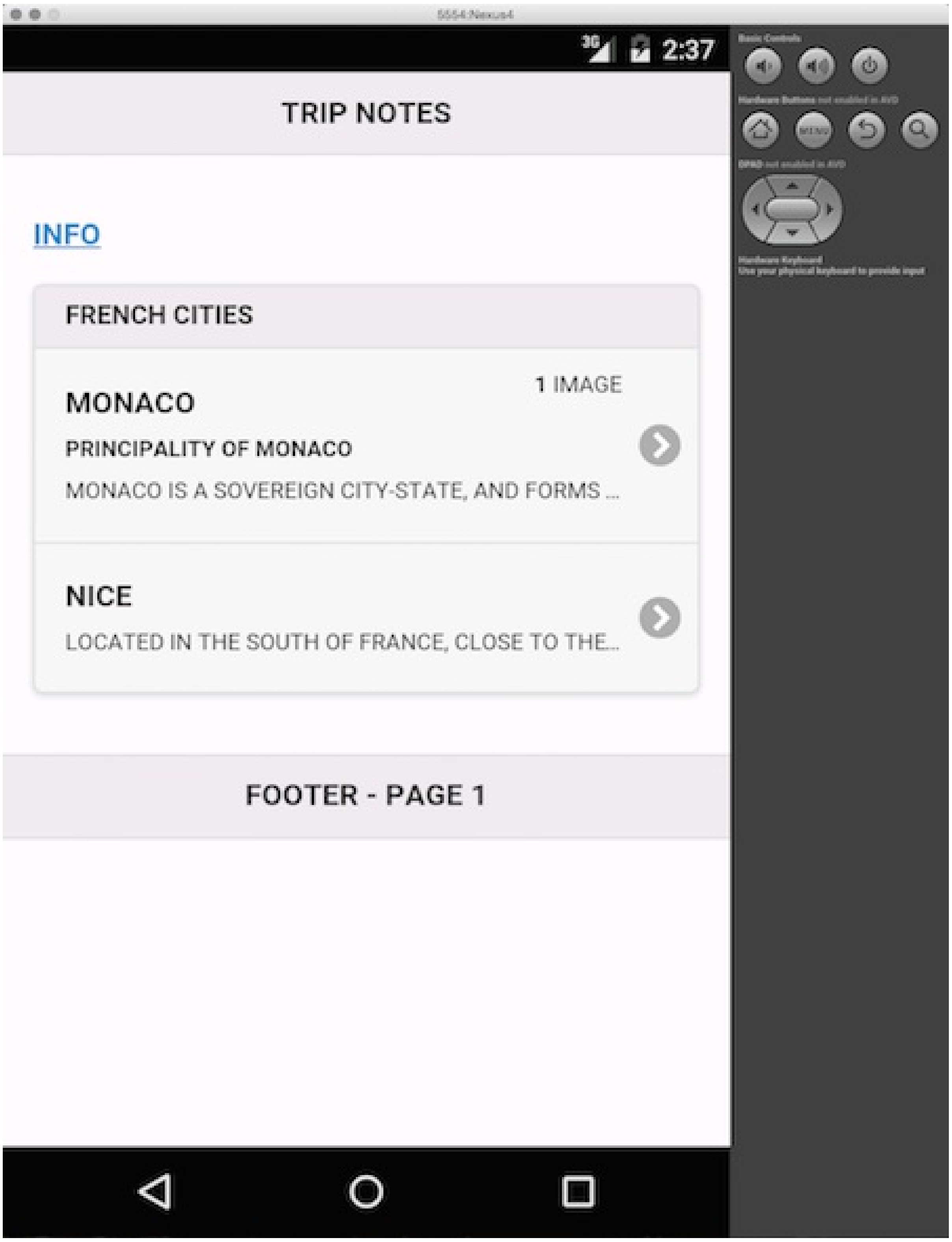


Image - Cordova app - Trip Notes - example 1


TRIP NOTES

INFO

FRENCH CITIES

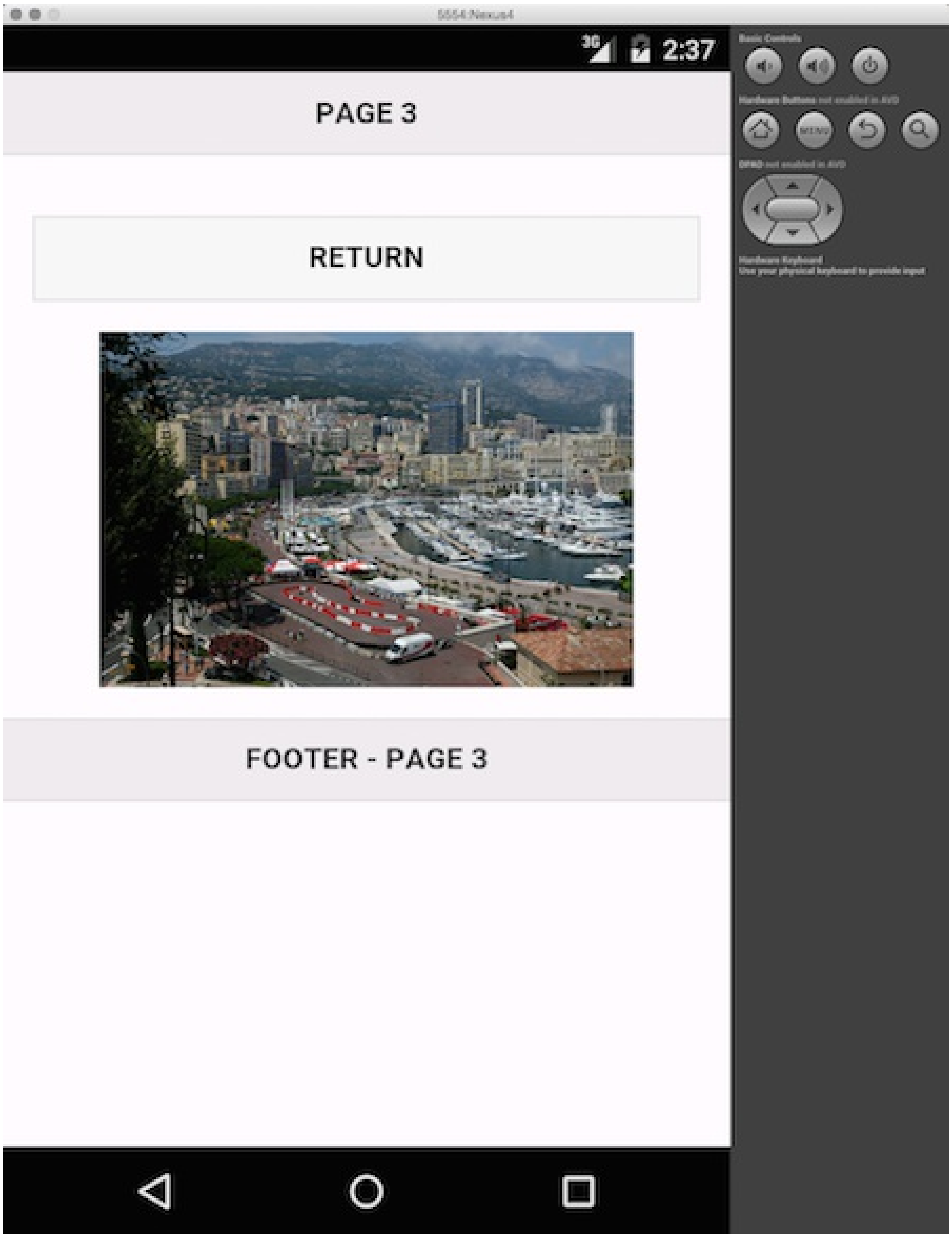
MONACO
1 IMAGE
PRINCIPALITY OF MONACO
MONACO IS A SOVEREIGN CITY-STATE, AND FORMS ...

NICE
LOCATED IN THE SOUTH OF FRANCE, CLOSE TO THE...

FOOTER - PAGE 1

5554:Nexus4
2:37
Basic Controls
Hardware Buttons not enabled in AVD
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input

Image - Cordova app - Trip Notes - example 2



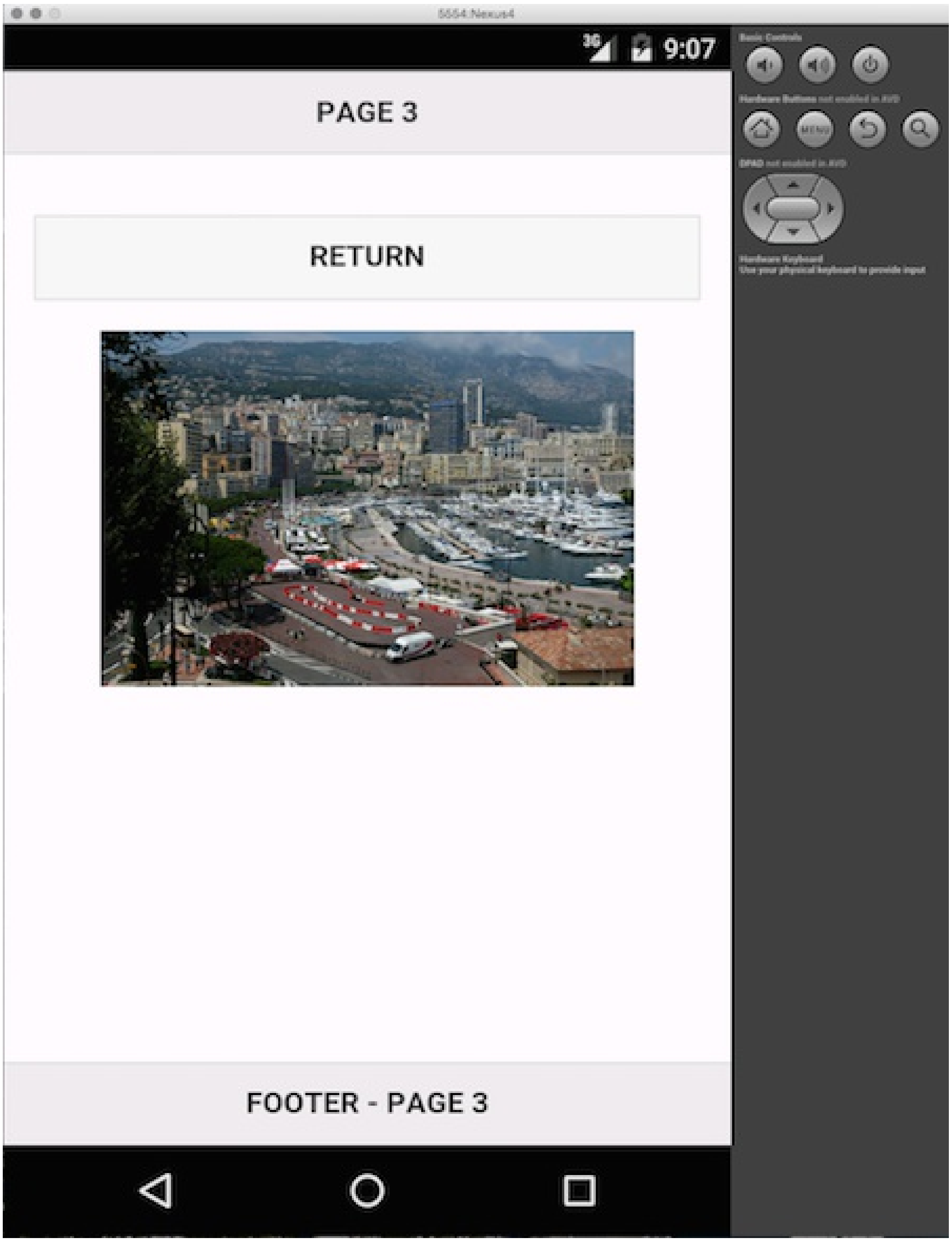
Cordova app - current design

Our current design includes a simple header, main, and a footer. To help with the rendering of our app's page, we can fix this footer to the bottom of the view by adding, for example, the following attribute in the HTML,

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

We can set this attribute on any of our footer sections in any of the views.

Image - Cordova app - Trip Notes - example 3


5554:Nexus4
3G
9:07
Basic Controls
Hardware Buttons not enabled in AVD
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input

PAGE 3

RETURN

FOOTER - PAGE 3