Comp 388/422 - Software Development for Wireless and Mobile Devices

Fall Semester 2015 - Week 9

Dr Nick Hayward

Next goals

- Cordova App
  - *API examples*
  - *add camera, gallery, geolocation...*
- Considering mobile design patterns

Cordova app - API plugin examples

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

Cordova app - API plugin examples - plugin test 2

setup

- create our initial plugin test shell application

```
cordova create plugintest2 com.example.plugintest2 plugintest2
```

- add any required plaforms, eg: Android, iOS, Windows Phone...

```
cordova platform add android
```

- 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 2

application structure

- now updated our initial Cordova template
  - *better structure for plugin test application*
  - *structure is now as follows*

```
|- hooks
|- platforms
   |- android
   |- platforms.json
|- plugins
   |- cordova-plugin-whitelist
   |- android.json
   |- fetch.json
|- resources
   |- icon
   |- splash
|- www
   |- assets
      |- images
      |- scripts
      |- styles
   |- docs
      |- json
      |- txt
      |- xml
   |- media
      |- audio
      |- images
      |- video
   |- index.html
|- config.xml
```

Cordova app - API plugin examples - plugin test 2

plugins - add camera plugin

- now add the camera plugin to our test application
- two ways we can add camera functionality to our application
  - *use the camera plugin*
  - *use the more generic Media Capture API*
- main differences include
  - ***camera*** *plugin focuses on camera capture and functionality*
  - ***media capture*** *includes additional options such as video and audio recording*
- add the camera plugin using the following Cordova CLI command

```
cordova plugin add cordova-plugin-camera
```

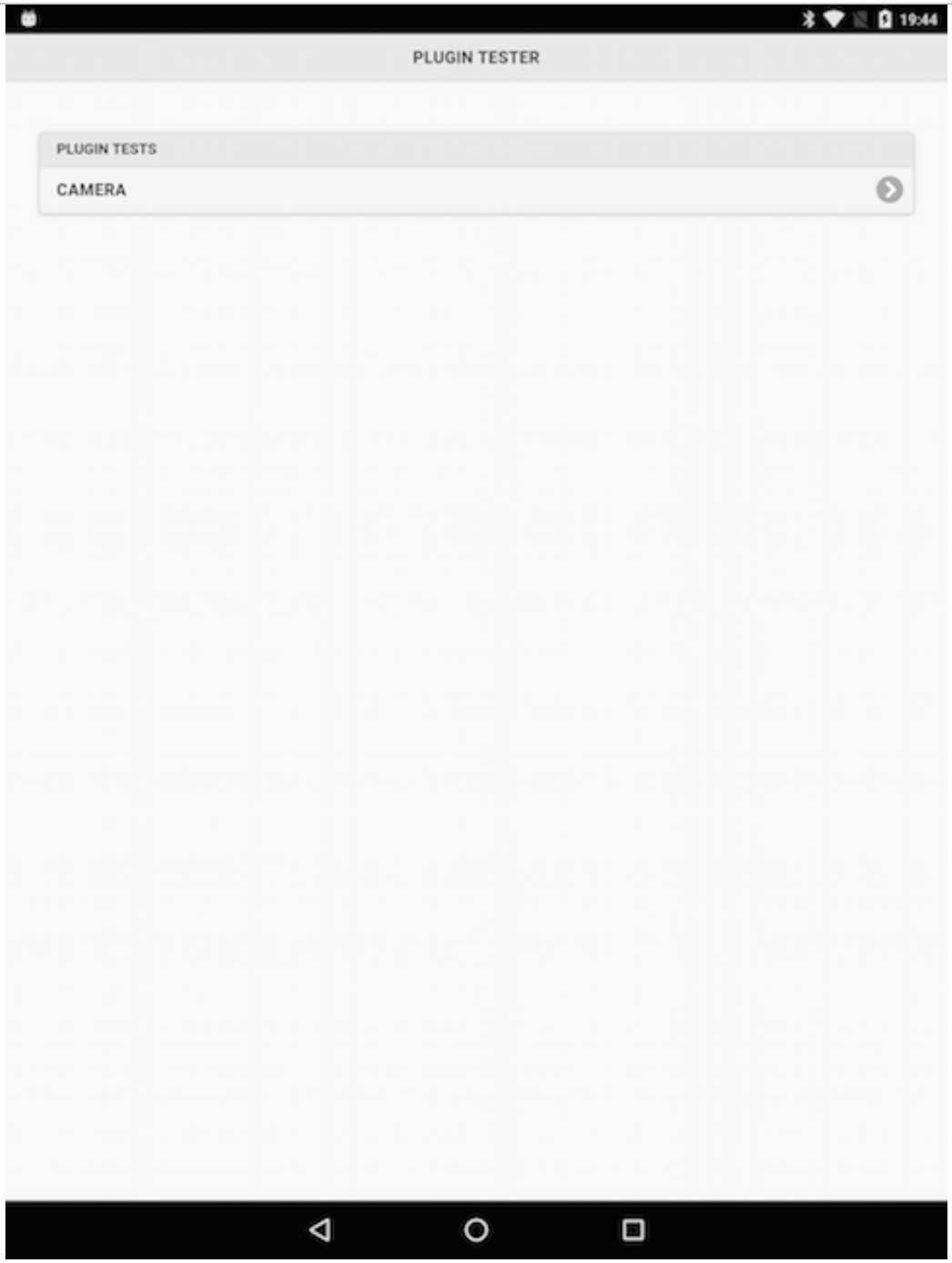
- provides standard `navigator` object
  - *enables taking pictures, and choose images from local image library*

Cordova app - API plugin examples - plugin test 2

plugins - add camera page

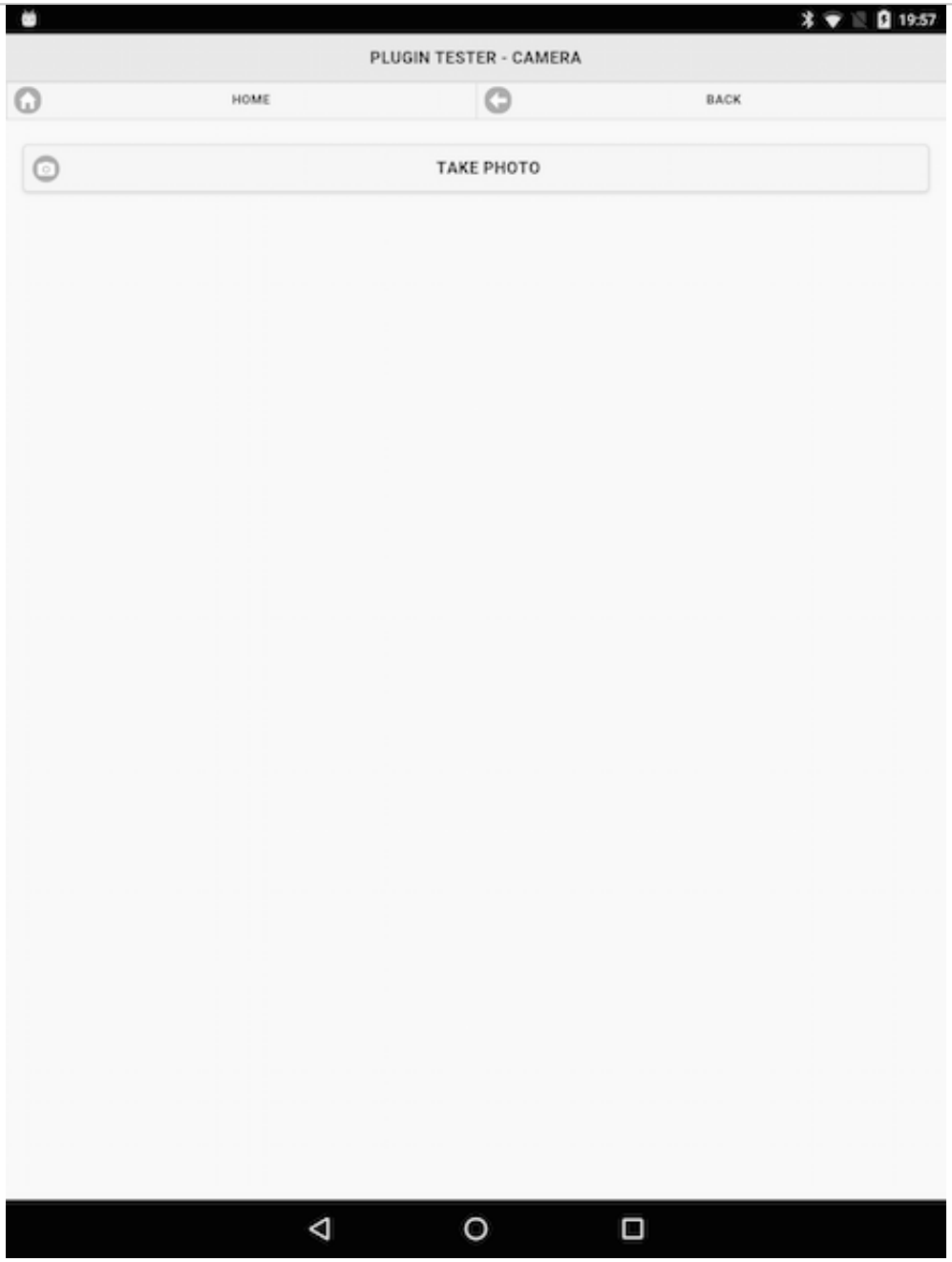
```
<!-- camera page -->
<div data-role="page" id="camera">
  <div data-role="header">
    <h3>plugin tester - camera</h3>
    </div><!-- /header -->
  <div data-role="navbar" data-iconpos="left">
    <ul>
      <li><a class="ui-btn" data-icon="home" data-transition="slide" href="#home">home</a>
      <li><a class="ui-btn" data-icon="arrow-l" data-rel="back">back</a></li>
    </ul>
  </div><!-- /navbar -->
  <div data-role="content">
    <input type="button" id="takePhoto" data-icon="camera" value="Take Photo" />
    <div id="photo">
      <img id="imageView" style="width: 100%;"></img>
    </div><!-- /photo -->
    <div data-role="popup" id="photoSelector" style="min-width: 250px;">
      <ul data-role="listview" data-inset="true">
        <li data-role="divider">Choose Photo</li>
        <li><a id="cameraPhoto" href="#">Take Photo with Camera</a></li>
        <li><a id="galleryPhoto" href="#">Get Photo from Gallery</a></li>
      </ul>
    </div><!-- /photoSelector -->
  </div><!-- /content -->
</div><!-- /camera page -->
```

Image - API Plugin Tester - Home



API Plugin Tester - initial home page

Image - API Plugin Tester - Camera



API Plugin Tester - initial camera page

Cordova app - API plugin examples - plugin test 2

plugins - add camera logic

- basic UI is now in place
- start to add some logic for taking photos with the device's camera
- need to be able to get photos from the device's image gallery
- app's logic in the `/assets/scripts/plugin.js` file
- handlers for the tap events
  - *a user tapping on the **takePhoto** button*
  - *then the options in the **photoSelector***
  - *take a photo with the camera*
  - *get an existing photo from the gallery*
- use the `onDeviceReady()` function
  - *add our handlers and processors for both requirements*
  - *add functionality for camera and gallery components*

Cordova app - API plugin examples - plugin test 2

plugins - add camera logic

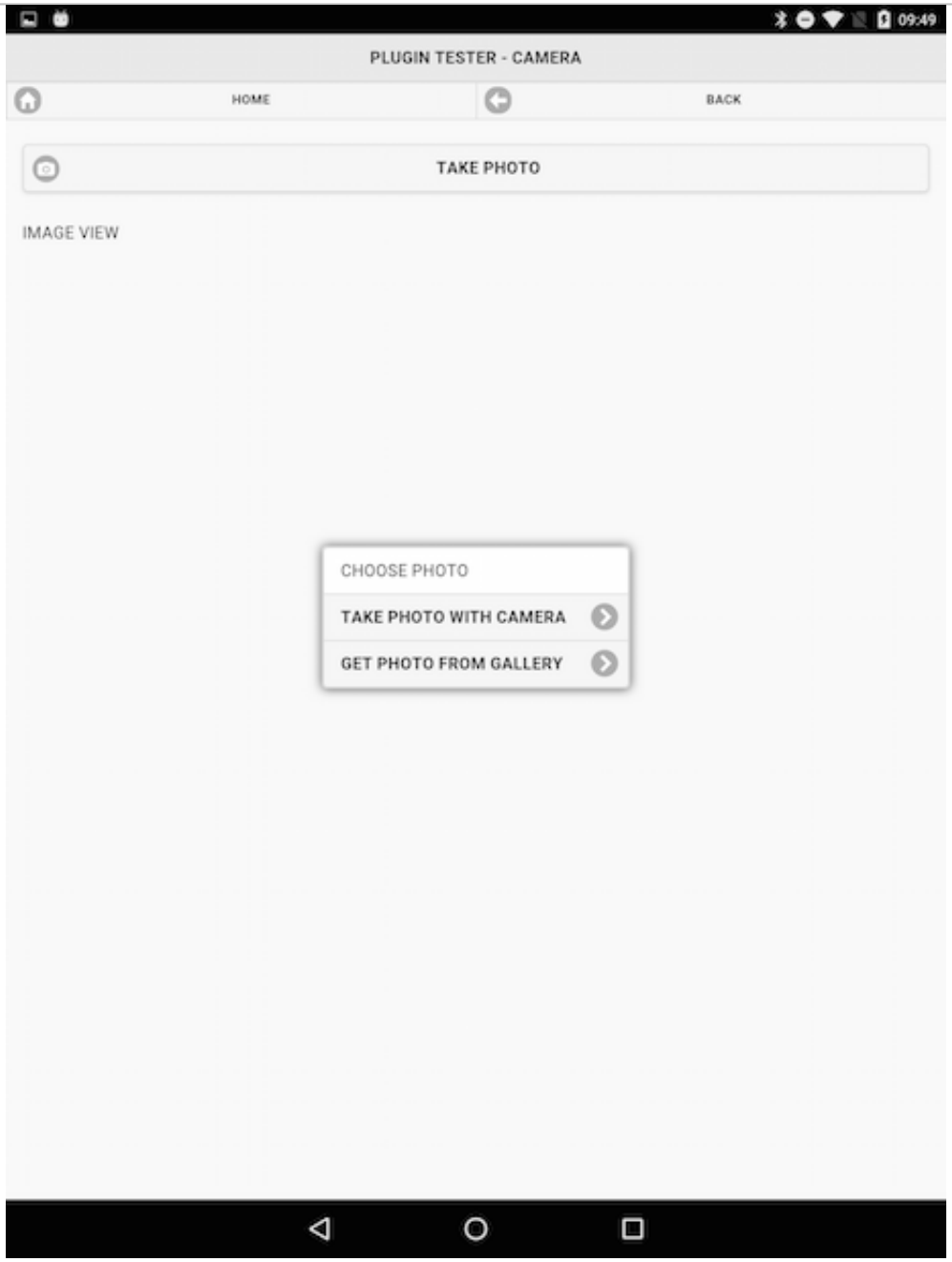
- add our handlers for the tap events
- initial handlers for `takePhoto`, `cameraPhoto`, and `galleryPhoto`

```
$("#takePhoto").on("tap", function(e) {
  e.preventDefault();
  //show popup options for camera
  $("#photoSelector").popup("open");
})

$("#cameraPhoto").on("tap", function(e) {
  e.preventDefault();
  //hide popup options for camera
  $("#photoSelector").popup("close");
})

$("#galleryPhoto").on("tap", function(e) {
  e.preventDefault();
  //hide popup options for camera
  $("#photoSelector").popup("close");
})
```

Image - API Plugin Tester - Camera



API Plugin Tester - camera photo selector

Cordova app - API plugin examples - plugin test 2

plugins - add camera logic

- capture an image using this plugin with the native device's camera hardware
- use the provided `navigator` object for the camera
  - *then call the `getPicture` function*
- also specify required callback functions for the camera
  - *and add some required options for quality...*

```
//Use from Camera
navigator.camera.getPicture(onSuccess, onFail, {
  quality: 50,
  sourceType: Camera.PictureSourceType.CAMERA,
  destinationType: Camera.DestinationType.FILE_URI
});
```

- quality option has been reduced to `50` for testing
  - *choose a value between 0 and 100 for our final application*
  - *100 being original image file from the camera*
- option for `destinationType` now defaults to `FILE_URI` could be changed to `DATA_URL`
  - ***NB:*** *`DATA_URL` option can crash an app due to low memory, system resources...*
  - *returns a base-64 encoded image*
  - *then render in a chosen format such as a JPEG*

Cordova app - API plugin examples - plugin test 2

plugins - add camera logic

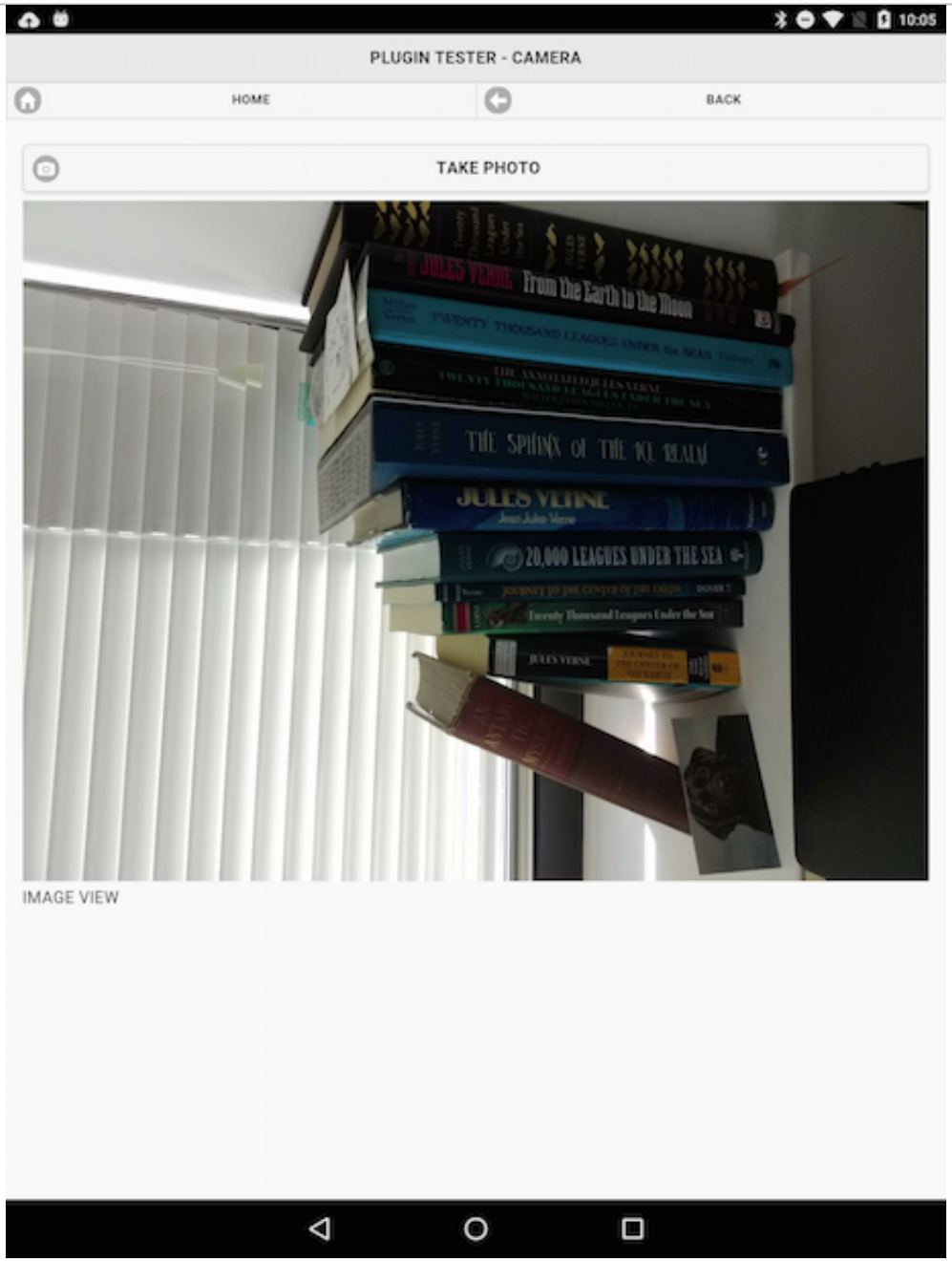
- two callback functions are `onSuccess` and `onFail`
  - *set logic for returned camera image and any error message*

```
function onSuccess(imageData) {
  //JS selector faster than jQuery...
  var image = document.getElementById('imageView');
  image.src = imageData;
}

function onFail(message) {
  alert('Failed because: ' + message);
}
```

- `onSuccess` function accepts a parameter for the returned image data
- using returned image data to output and render our image in the test `imageView`
- `onFail` function simply outputting a returned error message
- we can use these two callback functions to perform many different tasks
  - *we can pass the returned image data to a save function, or edit option...*
  - *they act like a bridge between our own logic and the native device's camera*

Image - API Plugin Tester - Camera



API Plugin Tester - image rotated

Cordova app - API plugin examples - plugin test 2

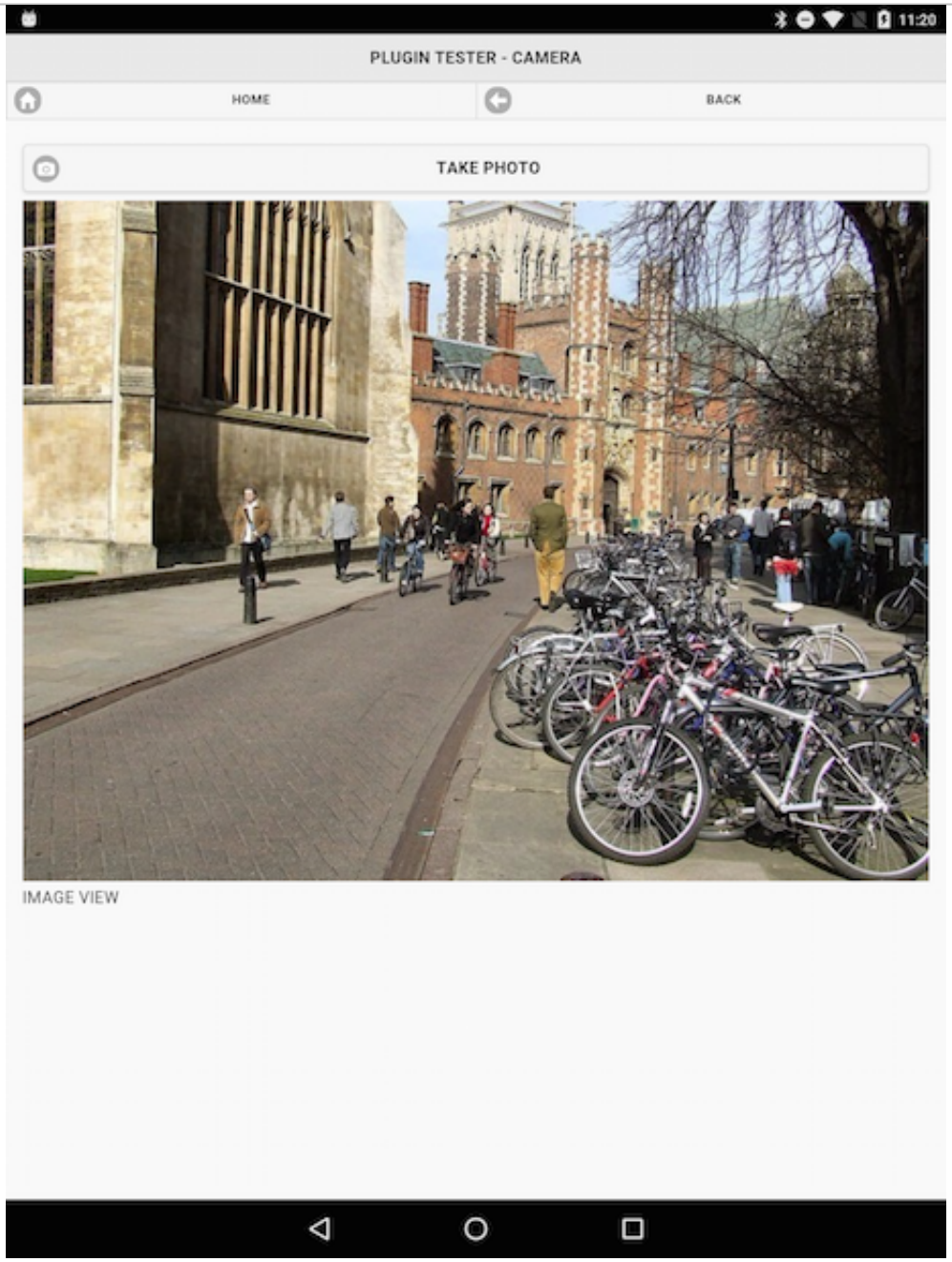
plugins - update camera logic

- returned an image from the camera
- update our application to select an image from gallery application
- add a conditional check to our `getPhoto()` function
  - *allows us to differentiate between a camera or gallery request*

```
navigator.camera.getPicture(onSuccess, onFail, {
  sourceType: Camera.PictureSourceType.PHOTOLIBRARY,
  destinationType: Camera.DestinationType.FILE_URI
});
```

- update in the `sourceType` from `CAMERA` to `PHOTOLIBRARY`
- returned image respects original orientation of gallery image

Image - API Plugin Tester - Camera



API Plugin Tester - image from gallery

Cordova app - API plugin examples - plugin test 2

plugins - fix camera logic

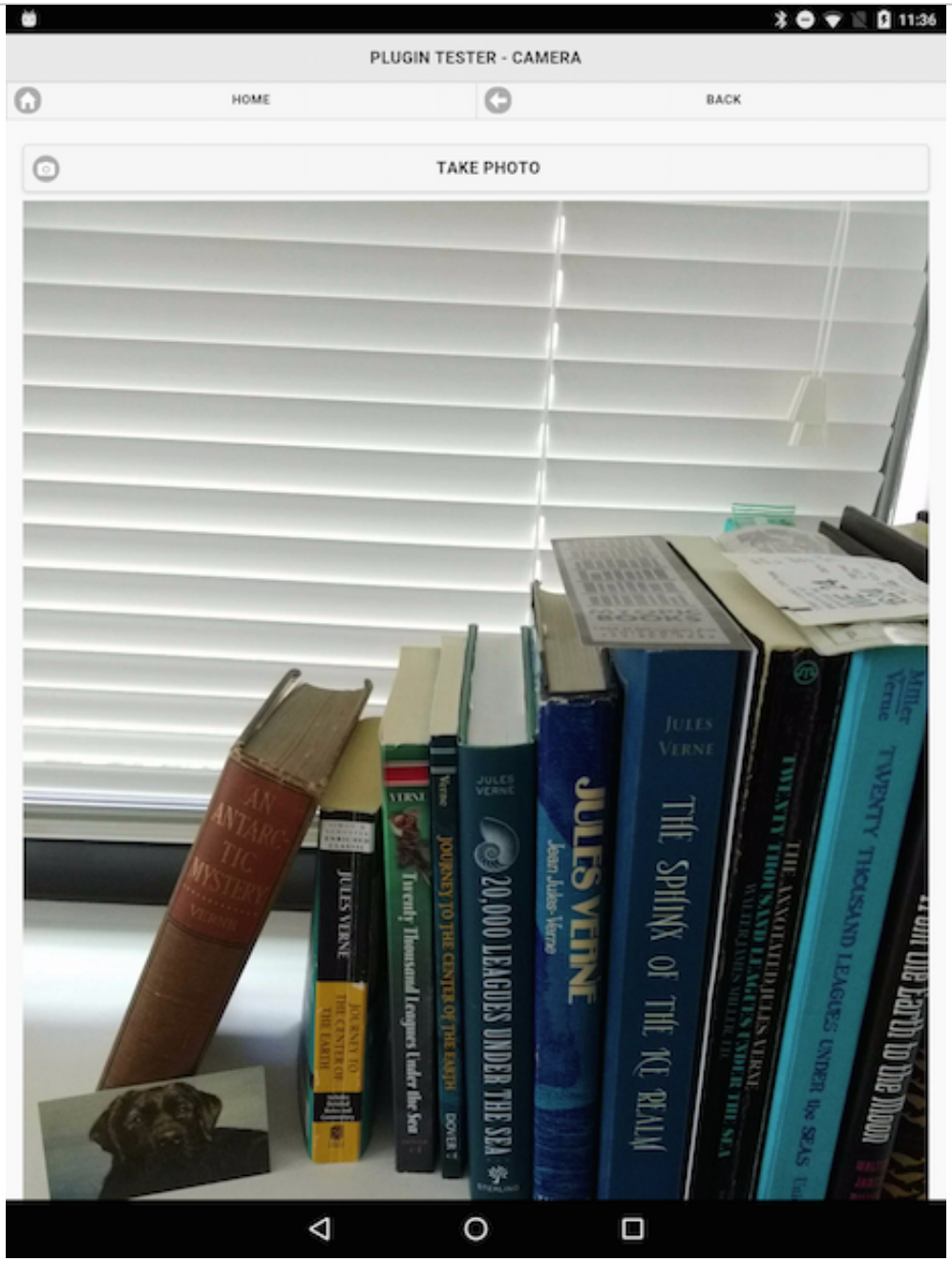
- need to fix the orientation issue with the returned image from the camera
- options for this plugin make it simple to update our logic for this requirement
  - *add a new option for the camera*

```
correctOrientation: true
```

- ensures that the original orientation of the camera is enforced
- updated logic is as follows

```
//Use from Camera
navigator.camera.getPicture(onSuccess, onFail, {
  quality: 50,
  correctOrientation: true,
  sourceType: Camera.PictureSourceType.CAMERA,
  destinationType: Camera.DestinationType.FILE_URI
});
```

Image - API Plugin Tester - Camera



API Plugin Tester - correct image orientation

Cordova app - API plugin examples - plugin test 2

plugins - camera updates

- continue to add many other useful options
  - *specifying front or back cameras on a device*
  - *type of media to allow*
  - *scaling of returned images*
  - *edit options...*
- in the app logic, also need to abstract the code further
  - *too much repetition in calls to the `navigator` object for the camera*
- then add more options and features
  - *save, delete, edit options*
  - *organise our images into albums*
  - *add some metadata for titles etc*
  - *add location tags for coordinates...*

Cordova app - API plugin examples - plugin test 3

plugins - geolocation

- add and use Cordova's **Geolocation** plugin
- helps us provide information about current location of user's device
- plugin returns data on device's location
  - *including latitude and longitude*
- plugin can use the following to help determine location
  - *GPS, network signals, phone network IDs...*
- API has been developed around the W3C's **Geolocation API Specification**
- **NB:** may not always be able to return a reliable location due to
  - *location restrictions*
  - *lack of access to a network*
  - *a user may reject location tracking and awareness...*
- need to be aware of potential privacy and security concerns
  - *application's privacy policy important*
  - *how we collect and whether we store data or not*
  - *how and when we share such data with 3rd-party services*
- consider offering user a simple opt-in/out option for location services
  - *app needs fallback options to cover lack of location services*

Cordova app - API plugin examples - plugin test 3

plugins - geolocation

- now create our test application for the **geolocation** plugin

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

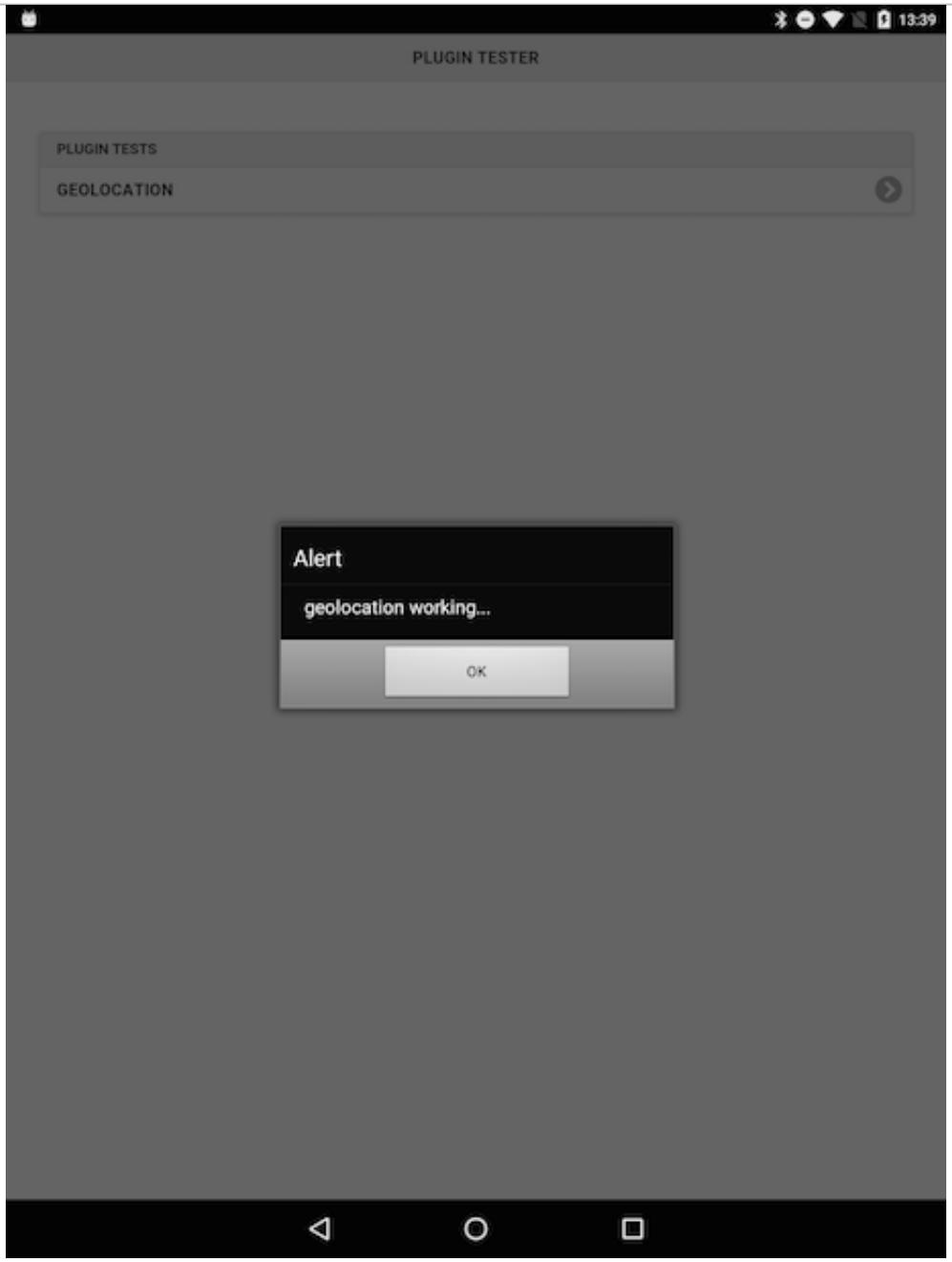
- update the www directory, modify initial settings in `config.xml`, and run initial test

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

- add **geolocation** plugin to our new project using the Cordova CLI

```
//cordova version 5.0+
cordova plugin add cordova-plugin-geolocation
//install directly via repo url
cordova plugin add https://github.com/apache/cordova-plugin-geolocation.git
```

Image - API Plugin Tester - Geolocation



API Plugin Tester - geolocation up and running

Cordova app - API plugin examples - plugin test 3

plugins - geolocation - test plugin

- add option to check and return current location of the user's device
- add a button to allow the user to request their current location
  - *then get the location's latitude and longitude*
  - *then output the location results to the user*

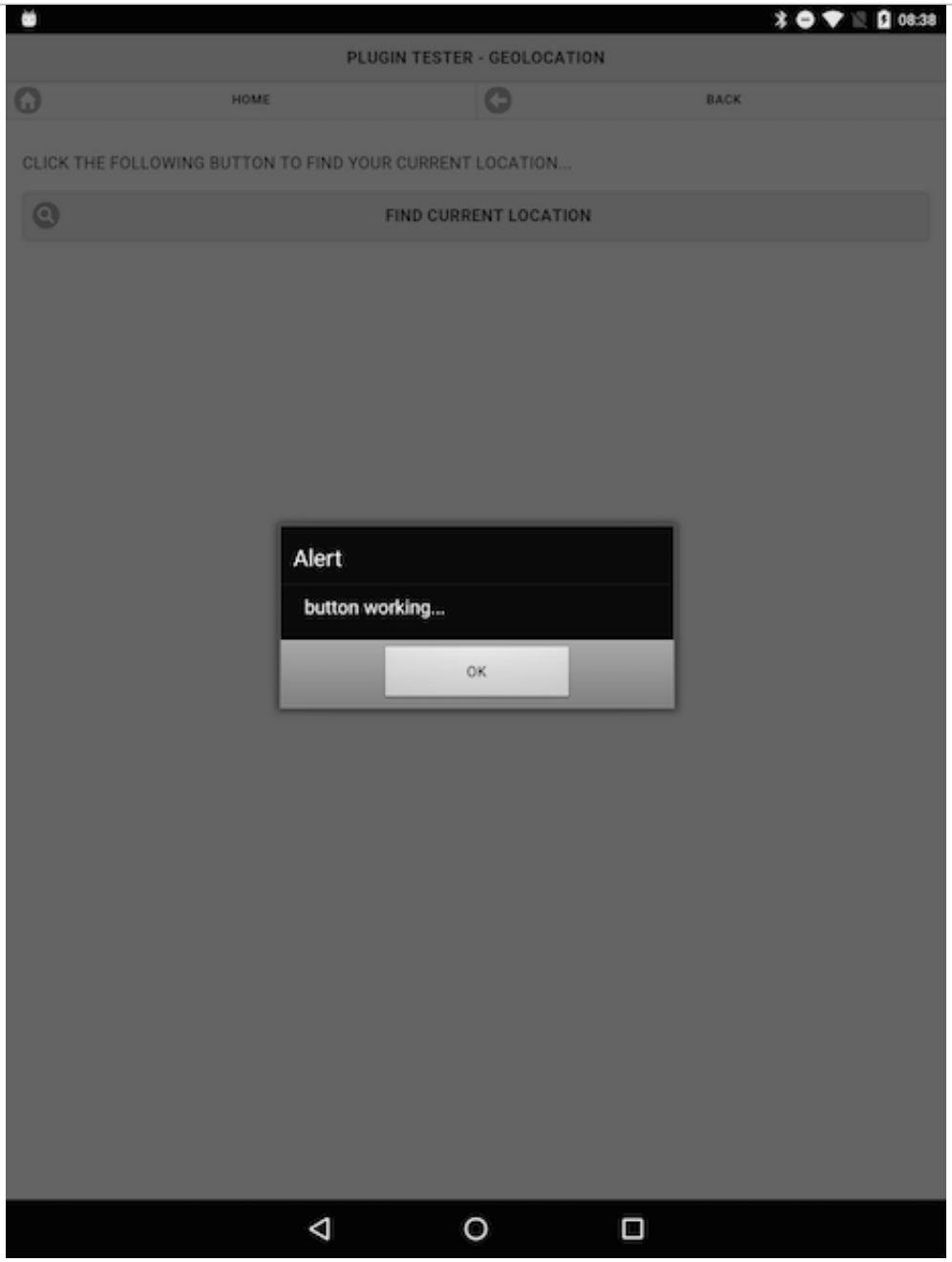
```
<div data-role="content">
  <p>Click the following button to find your current location...</p>
  <input type="button" id="getLocation" data-icon="search" value="Find Current Location" /
</div>
```

- then update the `plugin.js` file to handle the tap event for this button

```
//handle button press for geolocation
$("#getLocation").on("tap", function(e) {
  e.preventDefault();
    alert("button working...");
})
```

- output test alert for handler

Image - API Plugin Tester - Geolocation



API Plugin Tester - test geolocation button

Cordova app - API plugin examples - plugin test 3

plugins - geolocation - test plugin

- add our logic for working with the navigator object and the geolocation plugin
- first function we need to add is `getLocation()`
  - *use navigator object to get current position of user's device*
- add our standard success and fail callbacks
  - *initially add a timeout for poor signal or reception*
  - *enable high accuracy for this check*
  - *asking plugin to use most accurate source available, eg: GPS*
- `getLocation()` function is as follows,

```
function getLocation() {
  navigator.geolocation.getCurrentPosition(onSuccess,
    onFail, {
      timeout: 15000,
      enableHighAccuracy: true
  });
}
```

- standard callbacks for `onSuccess` and `onFail`

Cordova app - API plugin examples - plugin test 3

plugins - geolocation - test plugin

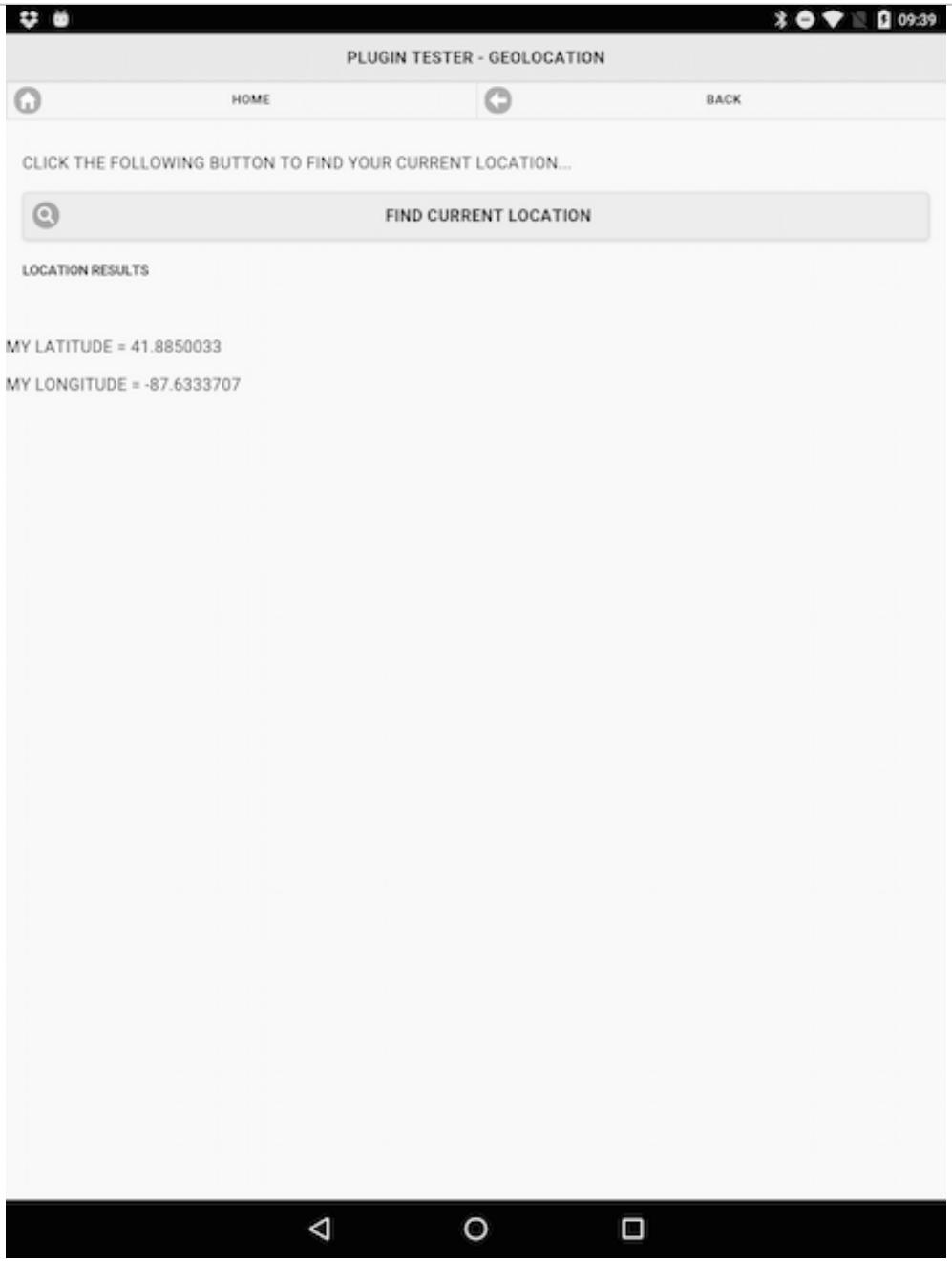
- successful return of location data
  - *use the latitude and longitude coordinates within our application*

```
function onSuccess(location) {
  var myLatitude = location.coords.latitude;
  var myLongitude = location.coords.longitude;
  //output result to #location div...
  $("#location").append("<p>my latitude = "+myLatitude+"</p><p>my longitude = "+myLongitud
}
```

- now store coordinates of user's location as latitude and longitude values
- various options for usage per application
  - *render to page, use with maps, add metadata to photos, track navigation...*
- also need to allow for the possibility of errors
  - *set our `onFail` callback as follows*

```
function onFail(error) {
  $("#location").append("location error code = "+error.code+" message = "+error.message);
}
```

Image - API Plugin Tester - Geolocation



API Plugin Tester - geolocation output

Cordova app - API plugin examples - plugin test 3

plugins - geolocation - plugin options

- additional options and properties available to us in the callbacks
  - *navigator object and properties for returned `location` object*
- add options to navigator object for geolocation
  - *`maximumAge` - cached position as long as it is not older than the specified age*
  - *age is specified as a number in milliseconds, eg: `maximumAge: 3000`*
- returned `location` object properties
  - ***altitude** - location.coords.altitude*
  - ***heading** - location.coords.heading*
  - ***speed** - location.coords.speed*
  - ***timestamp** - location.timestamp*
- fine-tune results for our users

Cordova app - API plugin examples - plugin test 3

plugins - geolocation - monitor location

- set plugin to monitor a device's location for changes

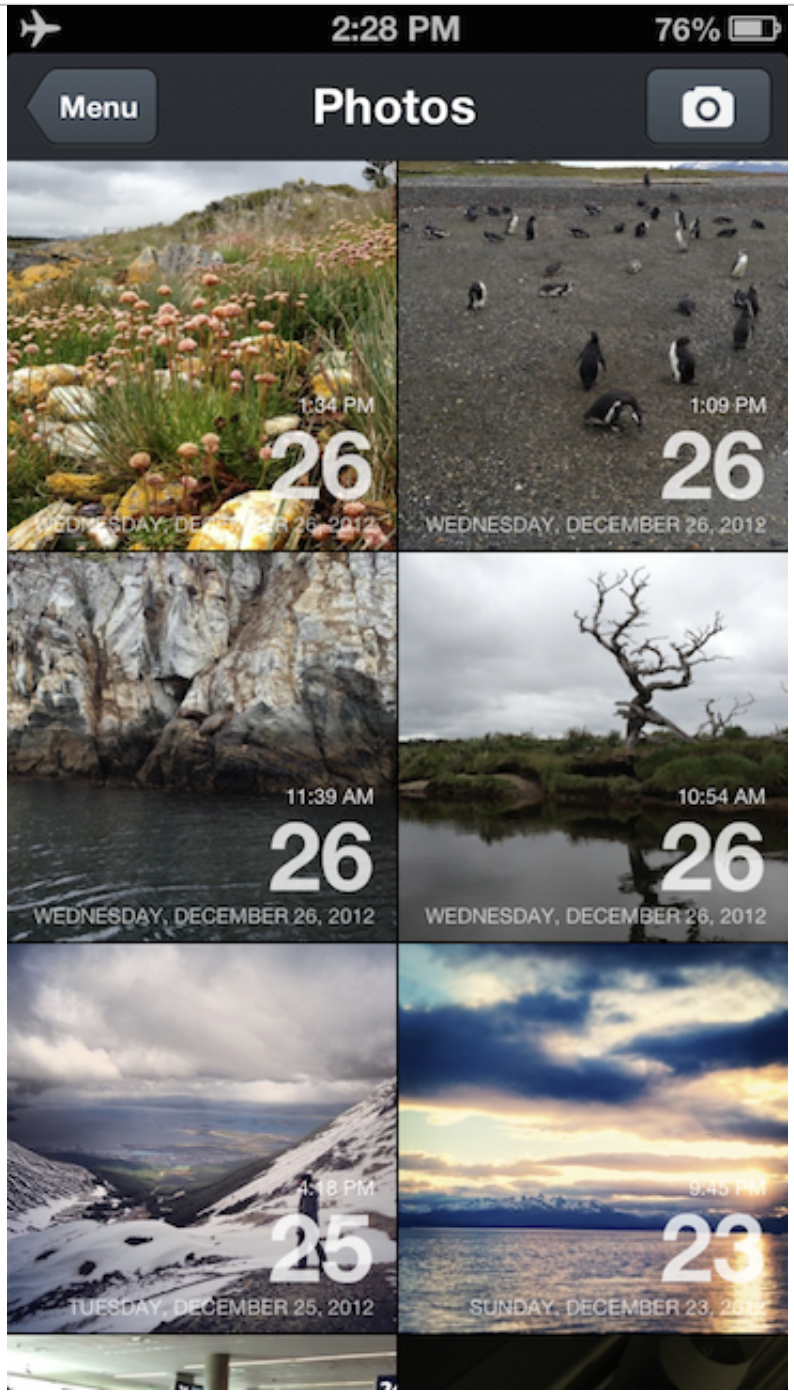
```
navigator.geolocation.watchPosition
```

- checking user's device for changes in their current location
  - *then returns device's location if a change is detected*

```
var monitorId = navigator.geolocation.watchPosition(onSuccess, onFail,
{option...}
);
```

- error callback and options are both optional
- also use returned ID with a `clearWatch()` function to stop ongoing location check and monitoring

Considering mobile design patterns - screen 1



Day One gallery on iOS

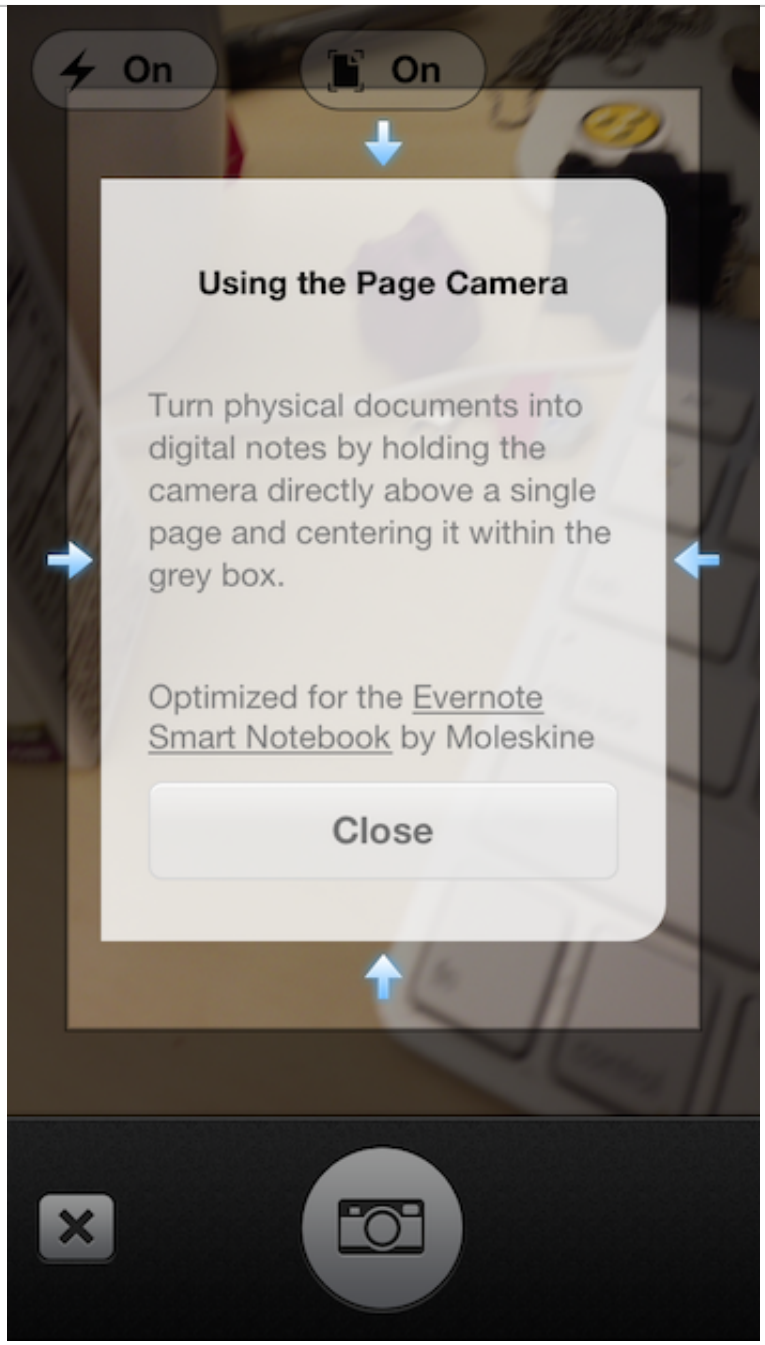
Considering mobile design patterns - screen 2

Flickr gallery on iOS

Considering mobile design patterns - screen 3

Facebook gallery on iOS

Considering mobile design patterns - screen 4

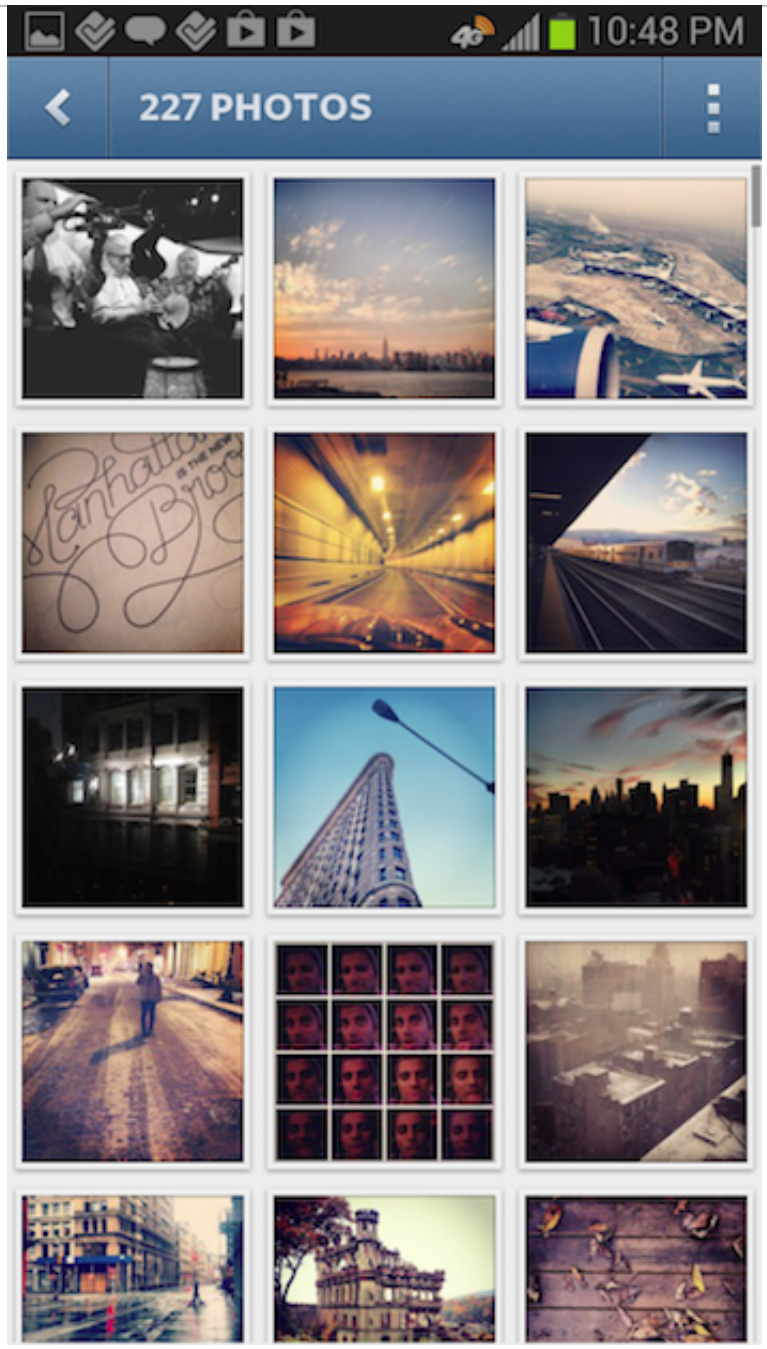


Evernote camera on iOS

Considering mobile design patterns - screen 5

Instagram camera on Android

Considering mobile design patterns - screen 6



Instagram gallery on Android

References

- Cordova
- Cordova API - camera plugin
- Cordova API - geolocation
- W3C
- Geolocation API Specification