Comp 422 - Software Development for Wireless and Mobile Devices

Fall Semester 2016 - Week 14

Dr Nick Hayward

Contents

Final Presentation & Report

- team presentation on 9th December @ 2.45pm
- team report due on 16th December by 2.45pm

Final Assessment Report

- report outline - demo and report

Extra demos & examples

- extra demos added to course's GitHub account
    - *source - 2016 - extras*
- cordova examples
    - *maptest*
    - *oauthtest*
    - *splitternav*
    - *sqltest*
- oauth
    - *google test with People API and user's profile*

Extra options - offline ready apps

- a few additional considerations as you prepare your app
    - *for users, testing, publication...*
- a consideration of offline support for our mobile app
- a mobile app needs to consider network usage
    - *with limited or no network connectivity*
    - *poor network reception*
    - *an explicit act by the user to restrict data usage*
- Cordova helps us prepare an app for offline usage
    - *bundles required files as it compiles the app*
- still many considerations for effective offline usage
    - *many disparate parts of our app affected by offline usage*
- not just issue with loss of connectivity to services, data, collaborative features &c
    - *also issue with UI design, interaction, and features*

Extra options - offline ready apps

- changes in state for an app element
- user offline unable to access end service for a request...
- as designers and developers
  - *need to be proactive in removing this option whilst offline*
  - *remove button &c. and option if network connectivity is lost*
  - *update element's state to `inactive` & modify interaction*
- act of updating state of an element for offline usage has a number of benefits
- with a disabled state
  - *visual rendering is updated correctly*
  - *event listeners should also become inactive*
  - *we remove any potential issues and errors*
  - *with app logic due to loss of connectivity*
- also offer feedback to the user to inform them
  - *why an element, option, or interaction is no longer available*
  - *inform them of the state of the network...*

Extra options - offline ready apps

- as our Cordova app loads
  - *set a listener for network related events*
- continue to check and monitor status of the network
- trigger changes in state as required during app's lifecycle
  - *app able to respond accordingly simply by checking `online` or `offline` state*
- need to monitor state of the app
  - *user may switch between states of network coverage and usage*
- Cordova provides useful **Network Information** plugin
- plugin has two notable features
  - *1. monitor type of connection our device is currently using*
  - e.g. `unknown`, `offline`, `wifi`, `4G` ...
  - *2. respond to events within our app for `offline` and `online`*
- use these events to modify our app and update feedback to users

Extra options - offline ready apps

- need to add the **Network Information** plugin,

```
cordova plugin add cordova-plugin-network-information --save
```

- then check standard `navigator` object for debice's connection type
- helps determine user's current connection, e.g. WiFi, 4G, &c
- by monitoring this connection type
  - *we can update our app's UI, interaction, and logic*
- start by adding necessary listeners for network state of our app

```
document.addEventListener("offline", offlineState, false);
document.addEventListener("online", onlineState, false);
```

- these event listeners monitor a change in our app's network status
  - *loss or gain of network connectivity triggers handler*
  - *change in connection type monitored*

Extra options - offline ready apps

- use custom functions
  - *update our app's UI for a disabled or enabled state*
  - *offer feedback to the user...*

```
//handle offline network state
function offlineState() {
  //handle offline network state
    console.log("app is now offline");
    //show ons alert dialog...
    ons.notification.alert('your app is now offline...');
}
```

```
//handle online network state
function onlineState() {
// Handle the online event
  var networkState = navigator.connection.type;
    console.log('Connection type: ' + networkState);
  if (networkState !== Connection.NONE) {
    //use connection state to update app, save data &c.
  }
  ons.notification.alert('Connection type: ' + networkState);
}
```

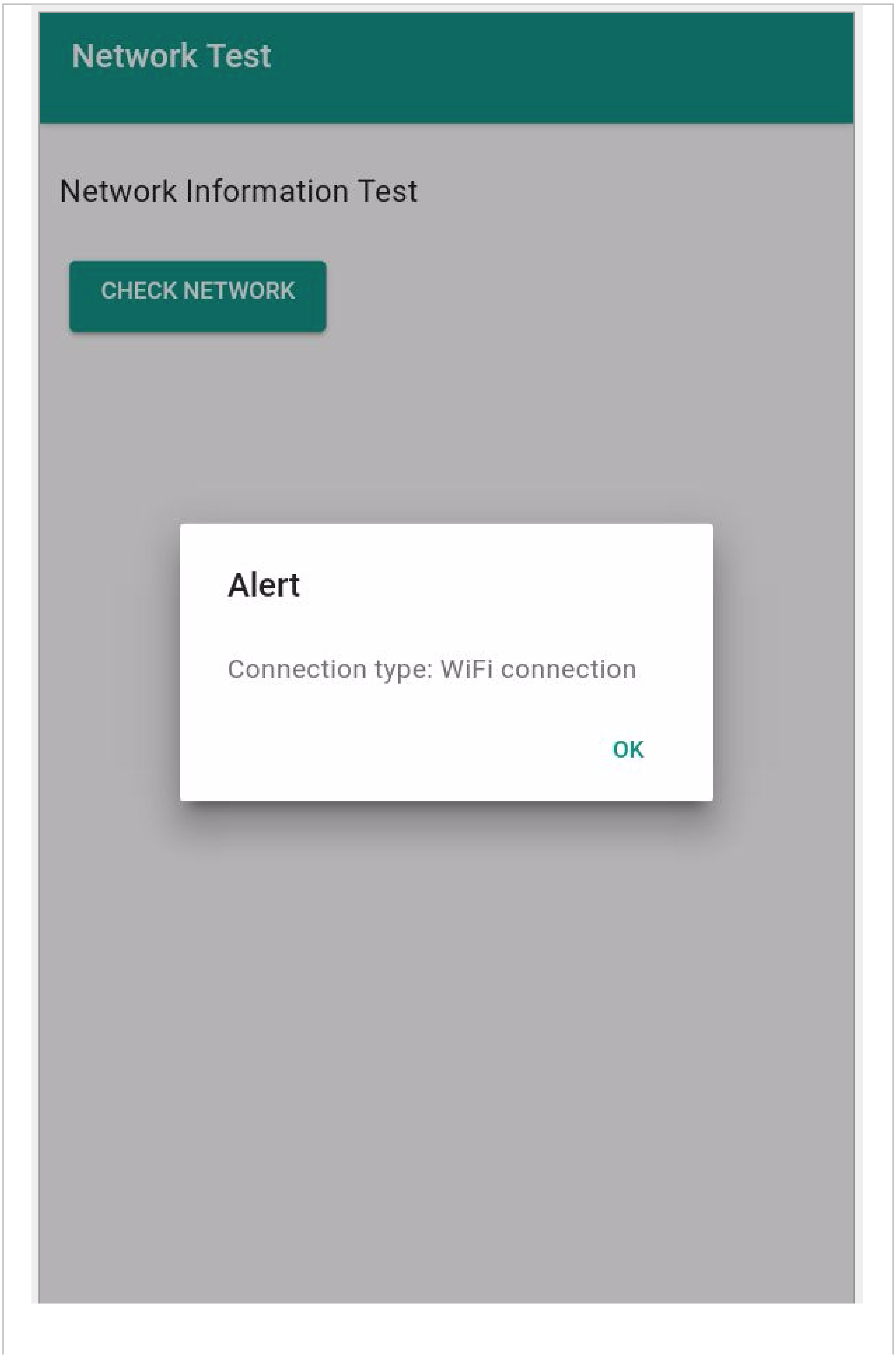
Extra options - offline ready apps

- quickly monitor and check our app's network status and type

```
function checkConnection() {
  var networkState = navigator.connection.type;
    console.log('check connection requested...');
  var states = {};
  states[Connection.UNKNOWN]  = 'Unknown connection';
  states[Connection.ETHERNET] = 'Ethernet connection';
  states[Connection.WIFI]     = 'WiFi connection';
  states[Connection.CELL_2G]  = 'Cell 2G connection';
  states[Connection.CELL_3G]  = 'Cell 3G connection';
  states[Connection.CELL_4G]  = 'Cell 4G connection';
  states[Connection.CELL]     = 'Cell generic connection';
  states[Connection.NONE]     = 'No network connection';

    console.log('Connection type: ' + states[networkState]);
}
```

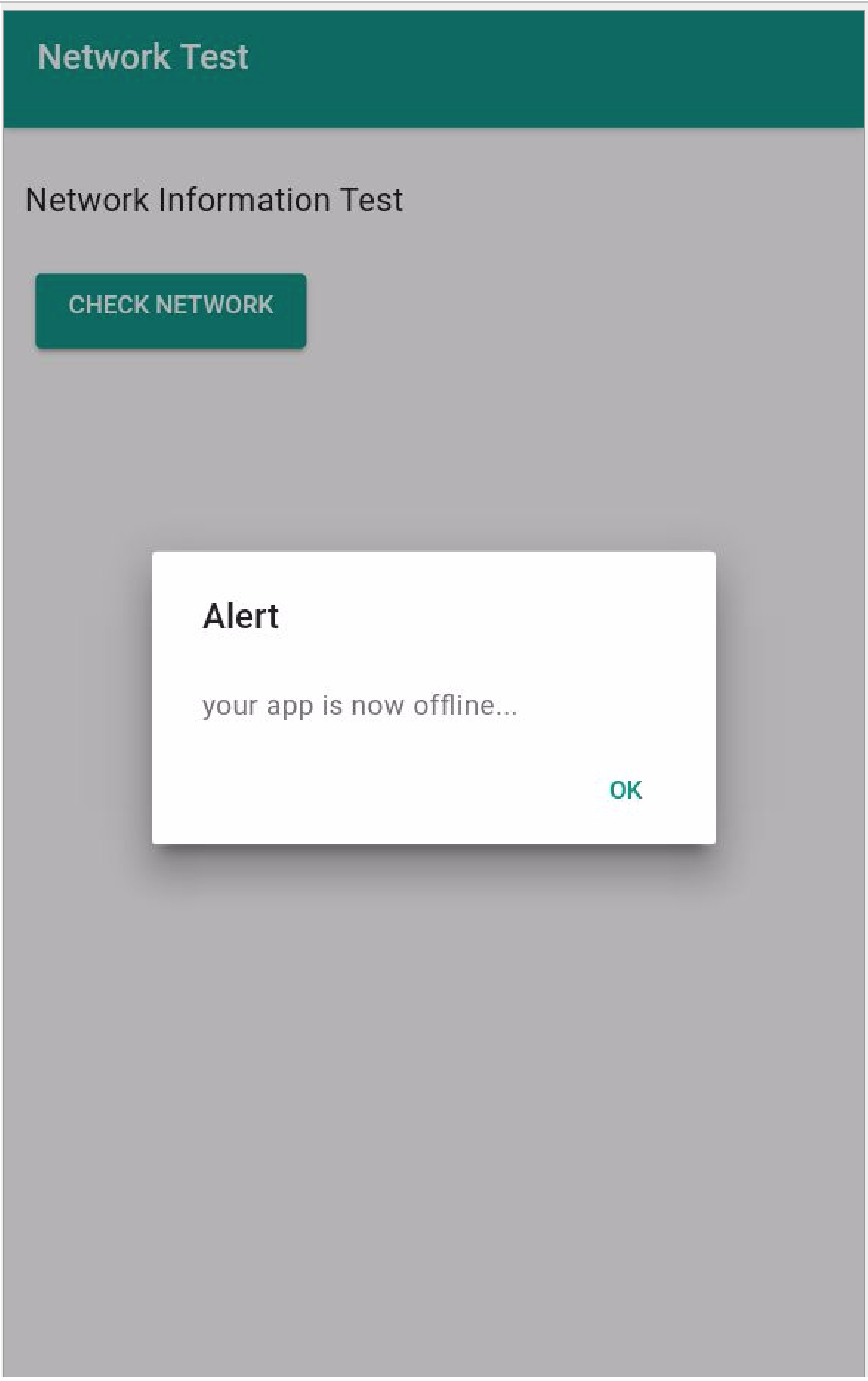
Image - Network Information - part 1



Connection type: WiFi connection

Network Test - connection type

Image - Network Information - part 2

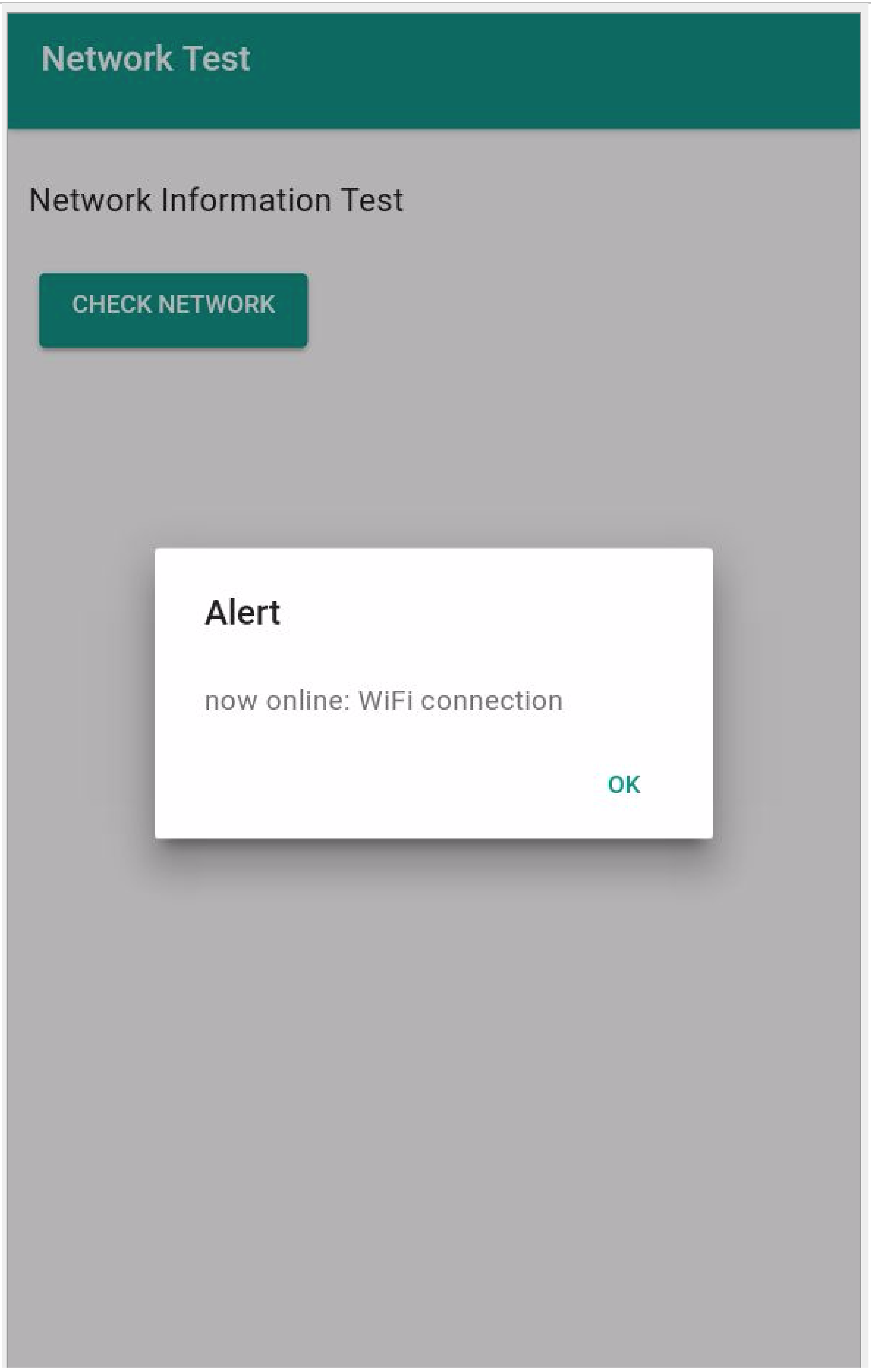


Alert

your app is now offline...

Network Test - check offline

Image - Network Information - part 3



Alert

now online: WiFi connection

Network Test - check online

Extra options - add International support

- consider publication and release for a mobile app
  - *need to remember international needs and preferences*
  - *different locales, languages, timezones...*
- use Cordova's **globalization** plugin

```
cordova plugin add cordova-plugin-globalization
```

- plugin uses a device's settings
  - *determines user's defined **locale**, **language**, and **timezone***
- e.g. user defined locale of **USA** & language setting of **UK English**
  - *apps will output dates, numbers, measures &c. in a USA compliant format*
  - *& render the language itself using UK English*

Extra options - add International support

- use this plugin with the defined global object
  - *after `deviceready` event*

```
navigator.globalization
```

- start by checking a user's defined language for the current app

```
navigator.globalization.getPreferredLanguage (
  //set success and error callbacks...
  function(language) {
    console.log('language = '+language.value);
  }, function() {
    console.log('error with language check...');
  }
);
```

- check a user's defined locale
  - *same pattern to language check...*

```
navigator.globalization.getLocaleName (
  //set success and error callbacks...
  function(locale) {
    console.log('locale = '+locale.value);
  }, function() {
    console.log('error with locale check...');
  }
);
```

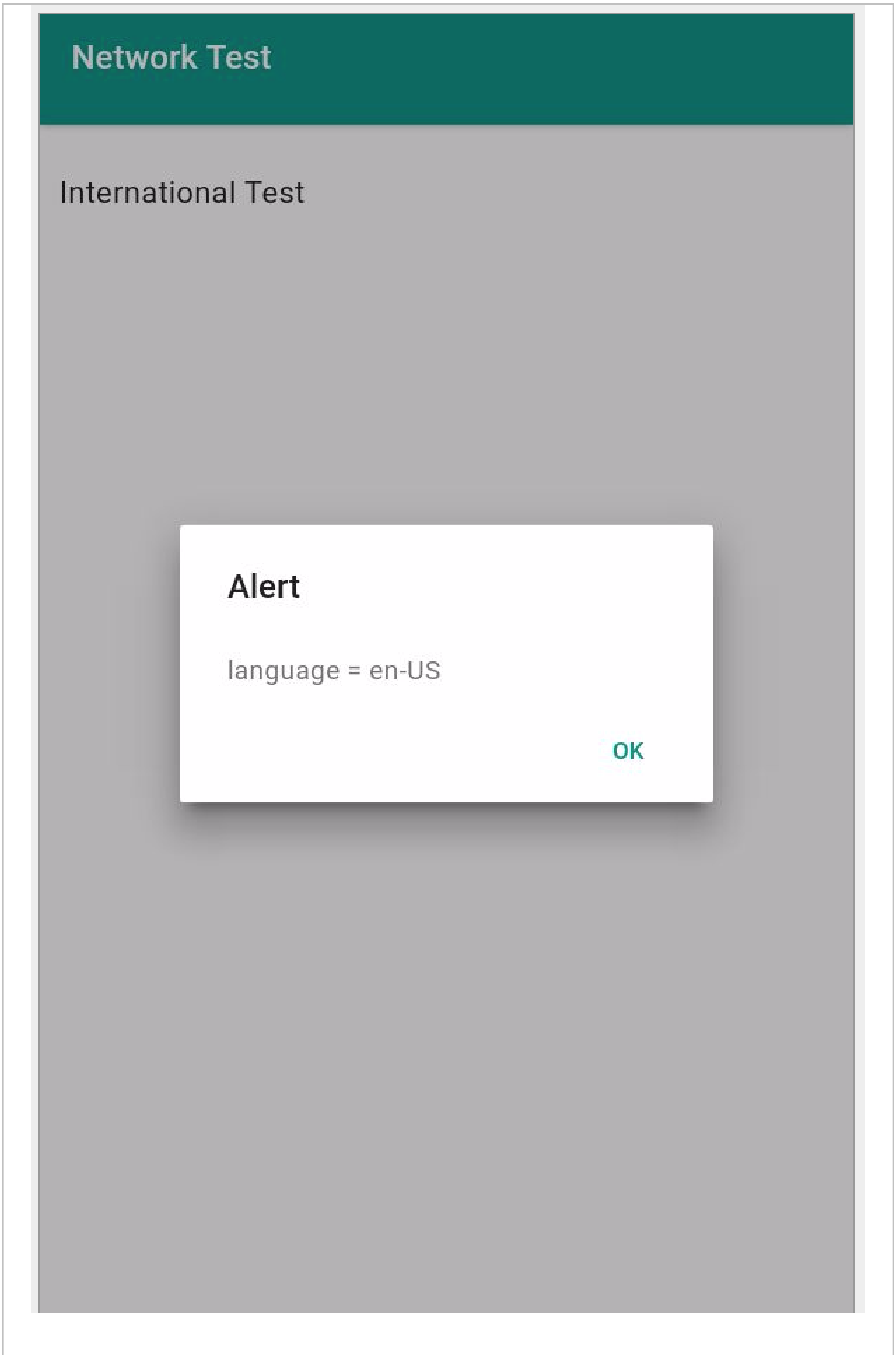
Extra options - add International support

- update and customise our app's dates and times
  - *correctly match the specified locale settings*
- use the `dateToString()` method with the navigator object

```
navigator.globalization.dateToString(
    new Date(),
    function (date) { alert('date: ' + date.value + '\n'); },
    function () { alert('Error getting dateString\n'); },
    { formatLength: 'short', selector: 'date and time' }
);
```

- example from the Cordova API docs - Globalization plugin
- date is created using JavaScript's `Date()` constructor
  - *then use it with `dateToString()` method on navigator object*
  - *ensure rendered date is formatted correctly to match set locale*

Image - International Support - part 1


Network Test

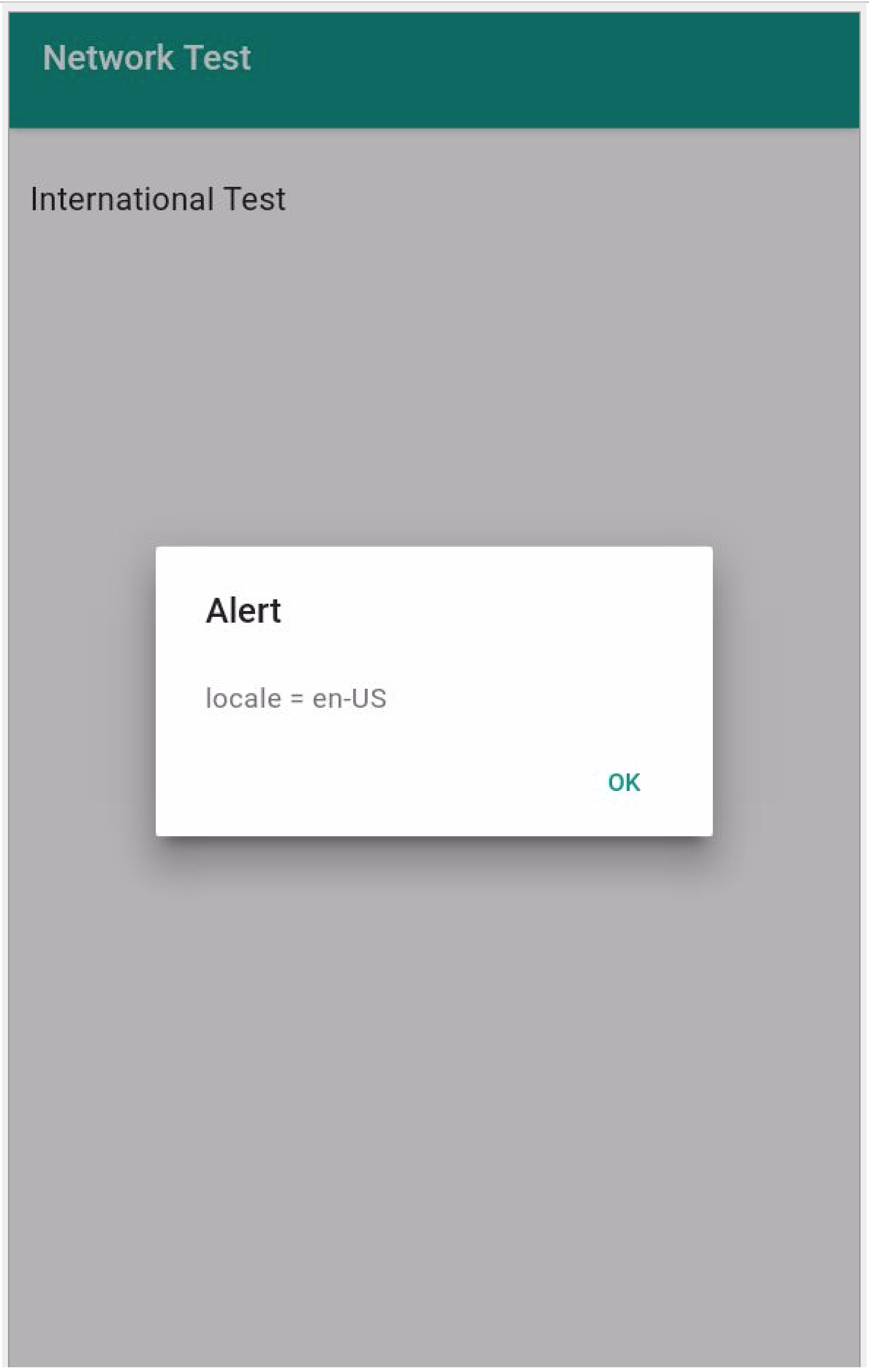
International Test

Alert

language = en-US

International support - language test

Image - International Support - part 2

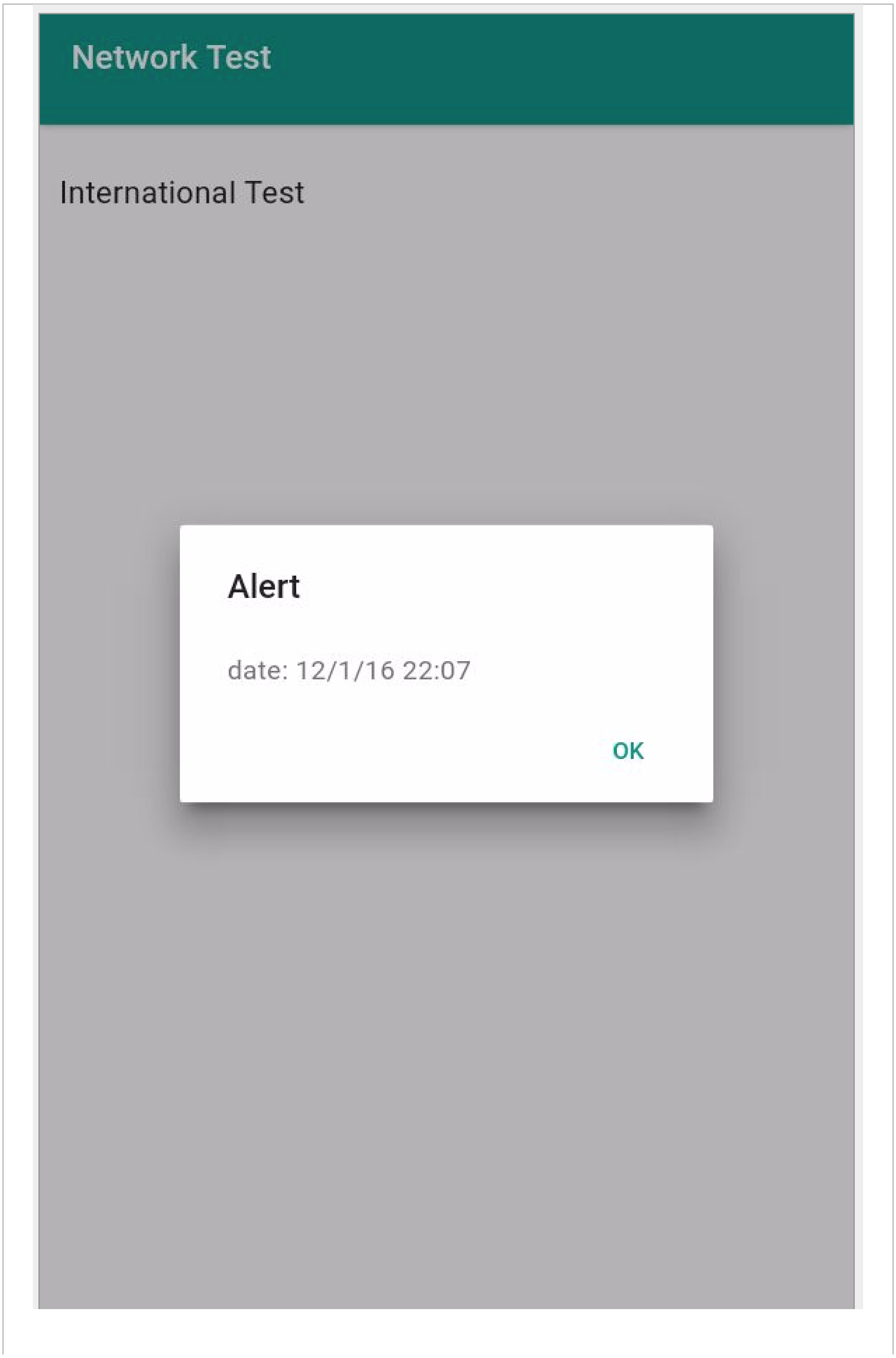


International Test

Alert

locale = en-US

Image - International Support - part 3



International Test

Alert

date: 12/1/16 22:07

International support - date and time test

Extra options - build and customisation - config.xml

- `config.xml` generated as part of Cordova CLI `create` command
- additional preferences we can consider in the metadata
- modify values of these preferences
  - *configure and setup our app with greater precision and customisation*
- Cordova uses `config.xml` file to help setup structures within an app
- standard metadata for author, description, app name, and ID
- additional, useful preferences, e.g.
  - *specifying the default start file as the app loads,*
  - *a security setting for resource access*
  - *a minimum API for building the app*
  - ...

Extra options - build and customisation - config.xml

- default start file will be specified as `index.html` in the config
- also update this value to a different file,

```
<content src="custom.html" />
```

- also update app's settings to define access privileges and domains for remote resources
  - *e.g. CSS stylesheets, JavaScript files, images, remote APIs, servers...*
  - *specifically remote resources that are not bundled with the app itself*
- Cordova refers to this setting as a **whitelist**
  - *now been moved to a specific plugin*
  - *added by default as we create an app*
- default value for this setting is global access, e.g.

```
<access origin="*" />
```

- this setting will be OK for many apps

Extra options - build and customisation - config.xml

- may need to restrict access, e.g.
  - *due to user input in our app*
  - *remote loading of data*
  - ...
- might consider restricting our app to specific domains
- add as many `<access>` tags as necessary for our app

```
<access origin="http://www.test.com" />
<access origin="https://www.test.com" />
```

- allows our app to access anything on this domain
  - *including secure and non-secure requests*
- also add subdomains relative to a given domain
  - *simply by prepending a wildcard option*

```
<access origin="http://*.test.com" />
<access origin="https://*.test.com" />
```

- we can now update our app to restrict access to specific, required domains
  - *e.g. remote APIs, servers hosting a DB...*

Extra options - build and customisation - config.xml

- also add further metadata and preferences to help customise our app
- already seen preferences for icons, splashscreens...
- also add further settings for
  - *plugins*
  - *specific installed and supported platforms*
  - *general preferences for all platforms*
  - *or restrict to a single platform*
- for general preferences there are five global options to consider, e.g.
  - *BackgroundColor*
  - *Android and iOS - specific fixed background colour*
  - *DisallowOverscroll*
  - *Android and iOS - prevent a rendered app from moving off the screen*
  - *Fullscreen*
  - *Android (but not iOS) - determine screen usage for an app*
  - *e.g. useful for kiosk style apps...*
  - *HideKeyboardFromAccessoryBar*
  - *iOS (but not Android) - hiding an additional toolbar above a keyboard*
  - *Orientation*
  - *Android (but not iOS) - locking an app's orientation*

Extra options - build and customisation - config.xml

- add any necessary preferences using the `<preference>` element in our `config.xml` file

```
<preference name="fullscreen" value="true" />
```

- add as many preferences as necessary for our app's configuration
- customise our preferences for a specific platform
  - *e.g. restricting a preference to just Android or iOS*

```
<platform name="android">
  <preference name="DisallowOverscroll" value="true" />
</platform>
```

Extra options - build and customisation - merge options

- many Cordova apps developed using a single code base
- with platform specific preferences and UI customisations
- may prefer to create a distinction in the app's design or functionality
- use **merges** options to create platform specific code, files...
- create a new folder called `merges` in our app's root directory
  - *not the www directory*
- use `merges` folder to add platform specific requirements
  - *e.g. css stylesheets*
- add sub-directory to `merges` for each supported platform
- when we build our Cordova app
  - *Cordova will check for a `merges` directory for each platform*
  - *files will replace existing in www directory*
  - *new files added to www directory*

```
config.xml
|-- hooks
|-- merges
```

```
    |__ android
    |__ ios
|-- platforms
|-- plugins
|-- www
```

Extra options - build and customisation - merge options

- example usage might include specific stylesheets per platform
- e.g. in our app's `index.html` file add a link to a CSS stylesheet
- stylesheet file added as usual to our app's `www` directory
  - *leave this CSS file blank for the overall project*
- then add matching CSS file to each platform directory in `merges` folder
- CSS file then added to our platform specific app as it is built by Cordova

```
config.xml
|-- hooks
|-- merges
    |__ android
        |__ css
            |__ platform.css
    |__ ios
|-- platforms
|-- plugins
|-- www
    |__ css
        |__ platform.css
    |__ ...
```

- allows us to add specific
  - *styling, layout, and design requirements*

  - *for each supported platform*
- quick and easy option for platform customisation

Extra options - build options - hooks

- we've been using Cordova's CLI tool to help
  - *create our apps, add platforms and plugins, build our apps...*
- we can customise the CLI tool using **hooks**
  - *scripts able to interact with the CLI tool for a given command and action*
- consider **Hooks** in two distinct scenarios
  - *before and after an action is executed by the CLI tool*
- for the CLI tool we might consider adding a **hook**
  - *before or after that command and action is called and executed*
- **hooks** might include automation of standard build options, tools, and commands
- e.g. automation of adding plugins to a project
  - *add a platform, and then add all required plugins using **hook***
- CLI tool checks for **hook** scripts in the `hooks` directory
- to add a **hook**
  - *create a sub-directory in the `hooks` directory - same name as a **hook***
  - *Cordova will then check for scripts to execute*
  - *scripts will be executed in alphabetical order by filename*
- **hooks** can be written in any language supported by the host computer

Extra options - prepare for release

- finalise our Cordova app
- need to consider preparation and packing of the app
  - *ready for publication to one or more app stores*
- each major app store conceptually follows a pattern for release
- to prepare our app for publication
  - *begin by transitioning app from development version to a stable release version*
  - *app requires signing by developer with password*
  - *define ownership of app*
  - *accept responsibility for publication, contents...*
- submit the app to a store for publication
  - *required to provide descriptions for the app itself*
  - *provide a minimum of screenshots for general usage and prominent features*
  - *add supplementary information for publication of app*

Extra options - prepare for release - Play Store

- releasing an Android app is considerably less involved than iOS
  - *developers can release and publish a vast array of application types*
- Play Store - division between preparation of the app, and then publication
- initial preparation
  - *begin by signing our app with a key - create using command line*
  - *use Cordova build tools to create a release build of our app*
- publication to store
  - *upload our app to Google's Play Store for publication*
  - *need to provide some additional supporting information*
  - *title for our app*
  - *icons*
  - *description*
  - *screenshots*
  - ...
  - *then mark our app as published*

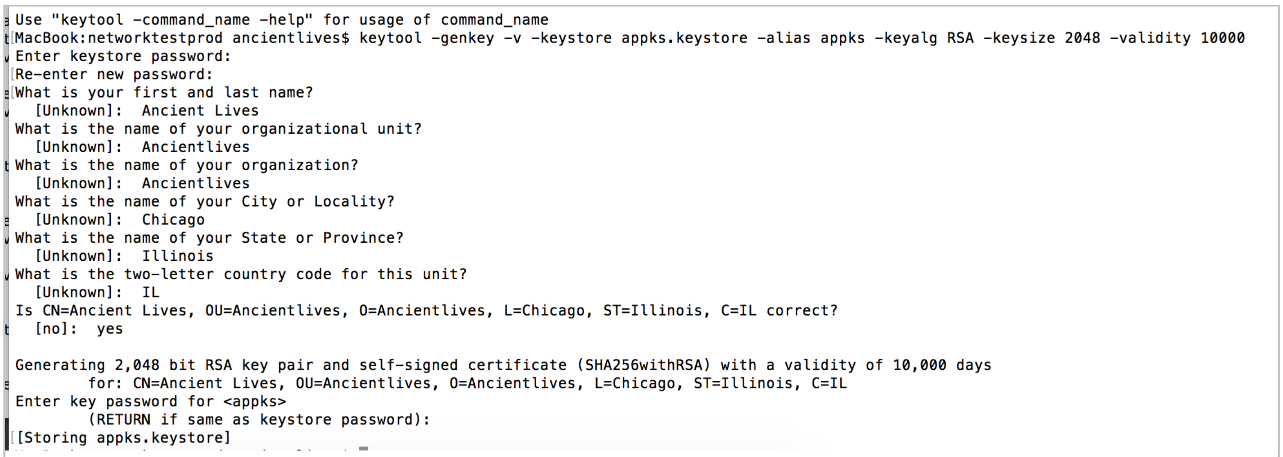
Extra options - prepare for release - signing

- prepare our app for a store
  - *need to sign it using a key store and key prior to publication*
  - *key signs the app, which is saved in the keystore*
- sign our app using the Java tool, **keytool**

```
keytool -genkey -v -keystore my-app-ks.keystore -alias my-app-ks -keya
```

- command creates both the keystore and key for our app
- command arguments to consider for `-keystore` and `-alias`
- `my-app-ks.keystore`
  - *filename for the keystore*
  - *can be set to a preferred name for your app*
- `my-app-ks`
  - *name of the alias for the keystore*
  - *developer can specify their preferred name*
  - *can be a simple, plain text name for the keystore*

Image - Keytool - Create a Keystore



```
Use "keytool -command_name -help" for usage of command_name
MacBook:networktestprod ancientlives$ keytool -genkey -v -keystore appks.keystore -alias appks -keyalg RSA -keysize 2048 -validity 10000
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  Ancient Lives
What is the name of your organizational unit?
  [Unknown]:  Ancientlives
What is the name of your organization?
  [Unknown]:  Ancientlives
What is the name of your City or Locality?
  [Unknown]:  Chicago
What is the name of your State or Province?
  [Unknown]:  Illinois
What is the two-letter country code for this unit?
  [Unknown]:  IL
Is CN=Ancient Lives, OU=Ancientlives, O=Ancientlives, L=Chicago, ST=Illinois, C=IL correct?
  [no]:  yes

Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 10,000 days
        for: CN=Ancient Lives, OU=Ancientlives, O=Ancientlives, L=Chicago, ST=Illinois, C=IL
Enter key password for <appks>
        (RETURN if same as keystore password):
[Storing appks.keystore]
```

Keytools - create a keystore

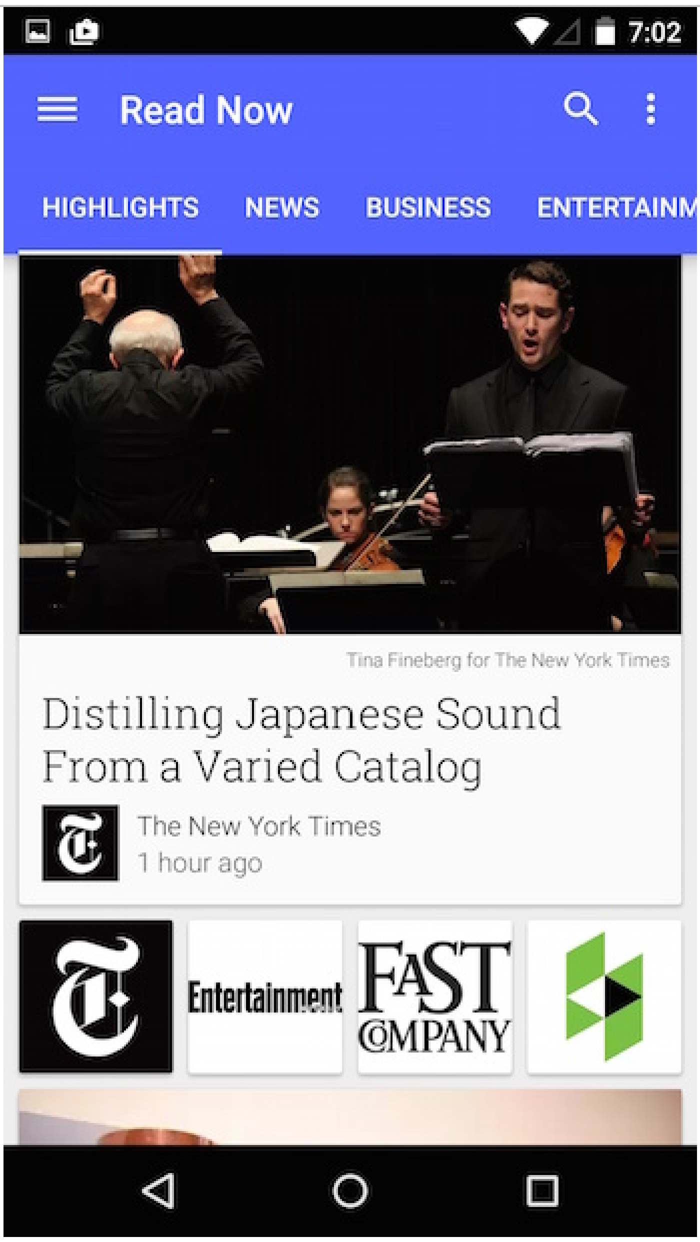
Considering mobile design patterns - discovery and browse

Consider discovery UI options, and designs that encourage or promote user browsing of an app and its content.

These include the following example types,

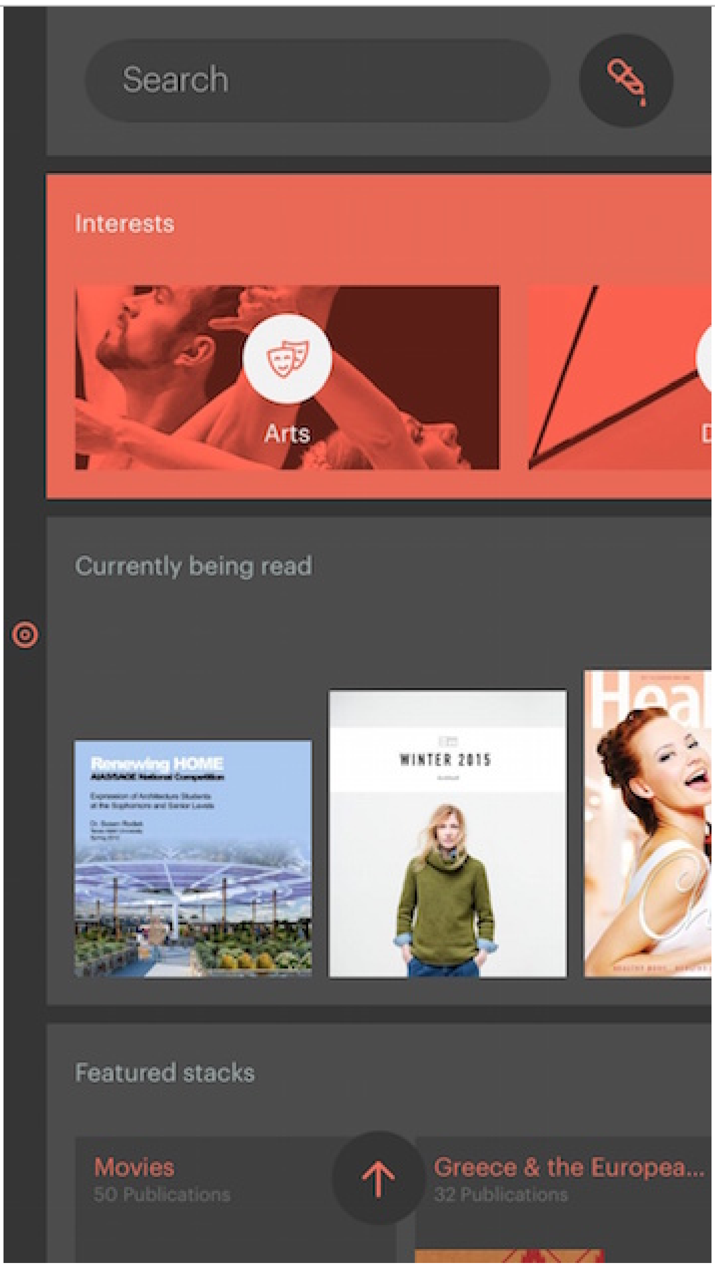
- discovery - screens 1 to 4
- coarch marks - screens 5 and 6
- filter options - screen 7
- maps - screen 8
- sidebars - screens 9 and 10

Screen 1 - Google Play Newstand on Android



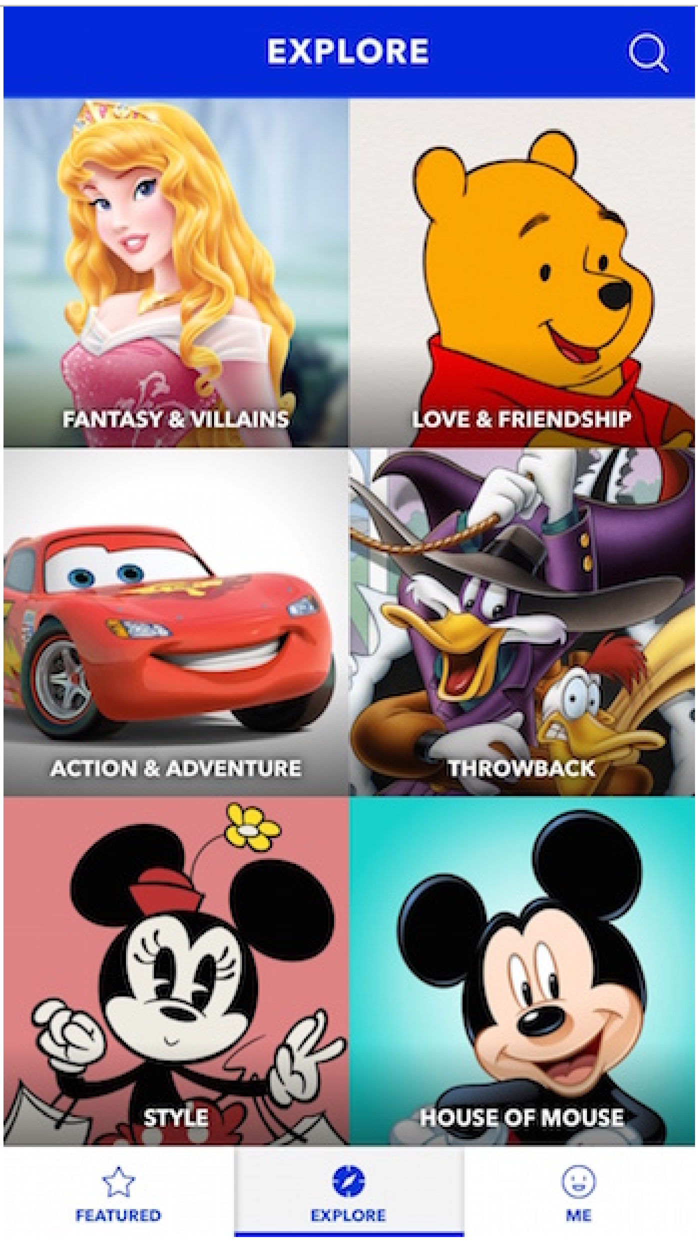
Google Play Newstand on Android

Screen 2 - Issuu magazine on iOS



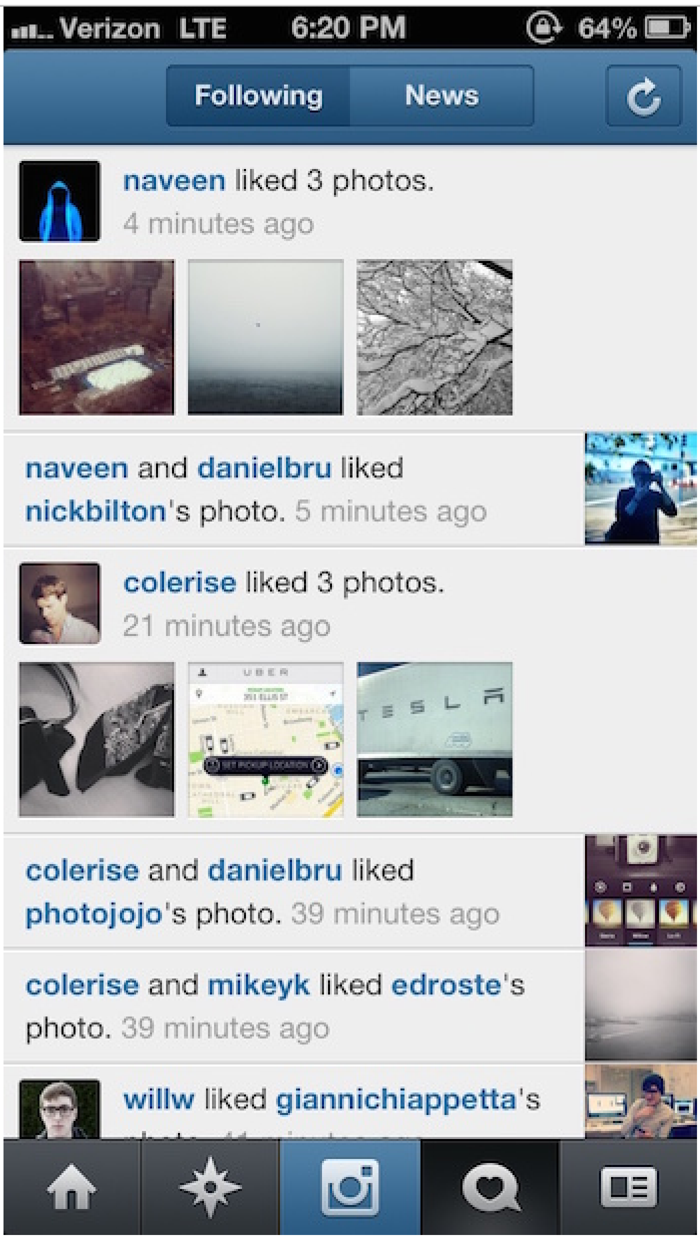
Issuu magazine on iOS

Screen 3 - Disney Inquizitive



Disney Inquizitive

Screen 4 - Instagram on iOS



naveen liked 3 photos.
4 minutes ago

naveen and danielbru liked nickbilton's photo. 5 minutes ago

colerise liked 3 photos.
21 minutes ago

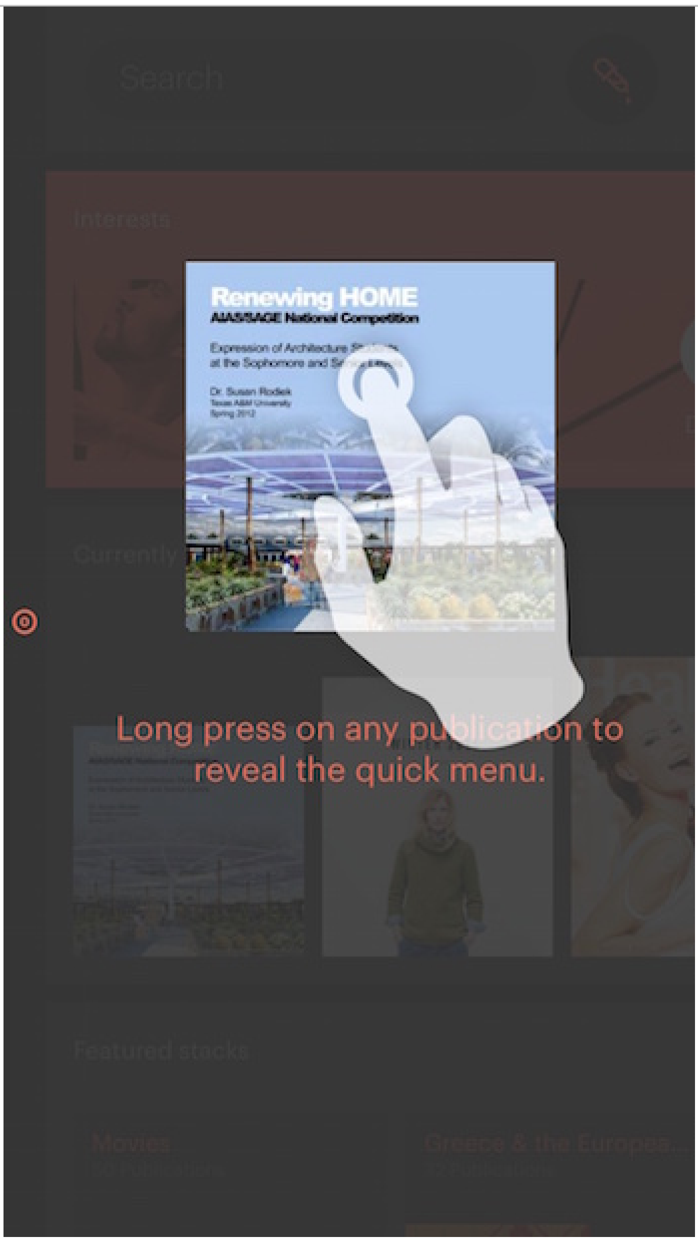
colerise and danielbru liked photojojo's photo. 39 minutes ago

colerise and mikeyk liked edroste's photo. 39 minutes ago

willw liked giannichiappetta's

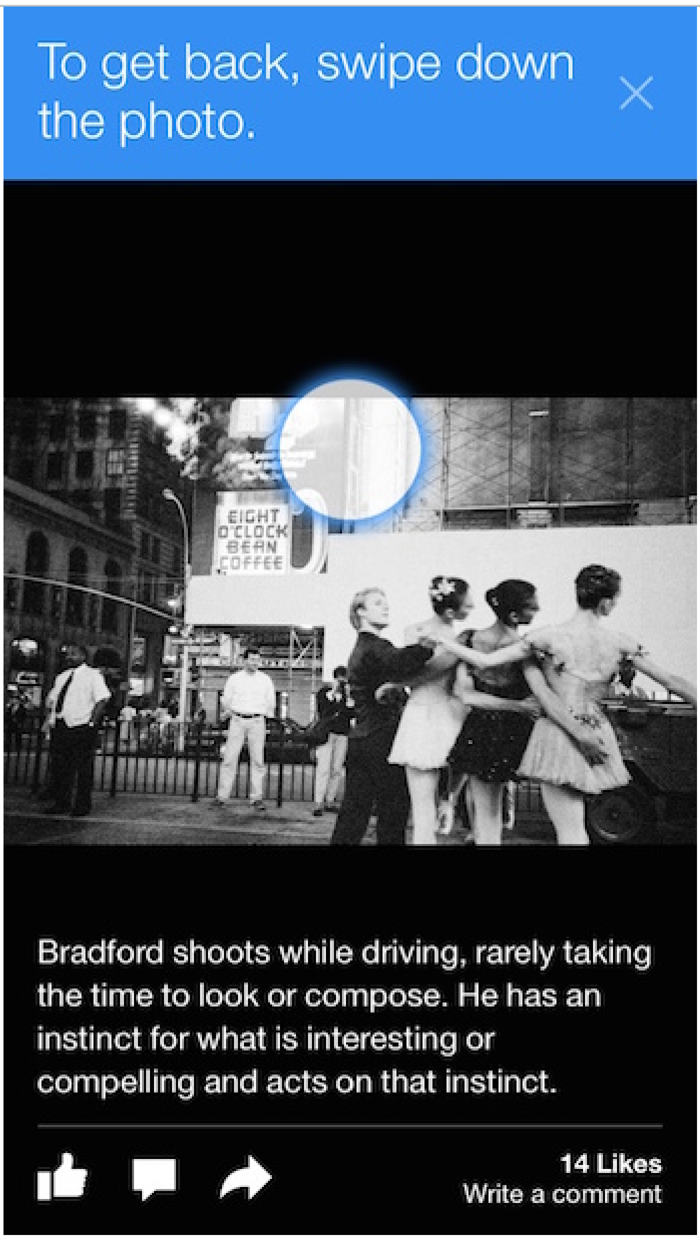
Instagram on iOS

Screen 5 - Issuu on iOS


Search
Interests
Renewing HOME
AIA/SAGE National Competition
Currently
Long press on any publication to reveal the quick menu.
Featured stacks
Movies
Greece & the Europea...

Issuu on iOS

Screen 6 - Paper (stories from Facebook)

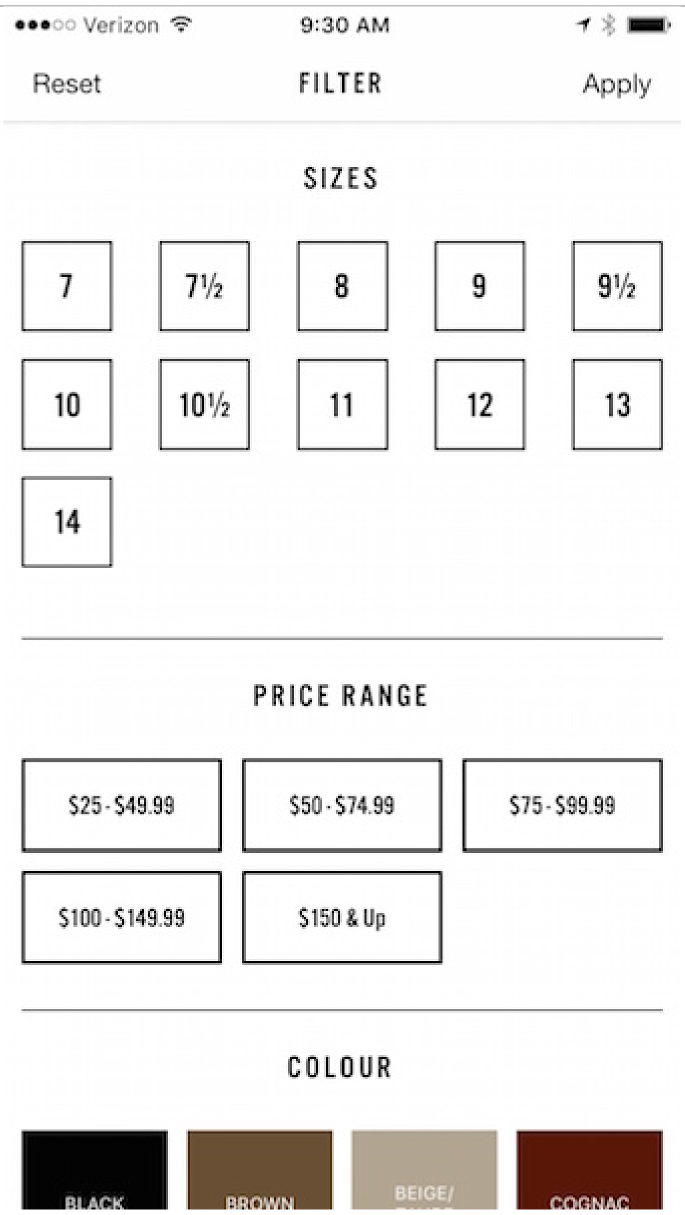

To get back, swipe down the photo.

Bradford shoots while driving, rarely taking the time to look or compose. He has an instinct for what is interesting or compelling and acts on that instinct.

14 Likes
Write a comment

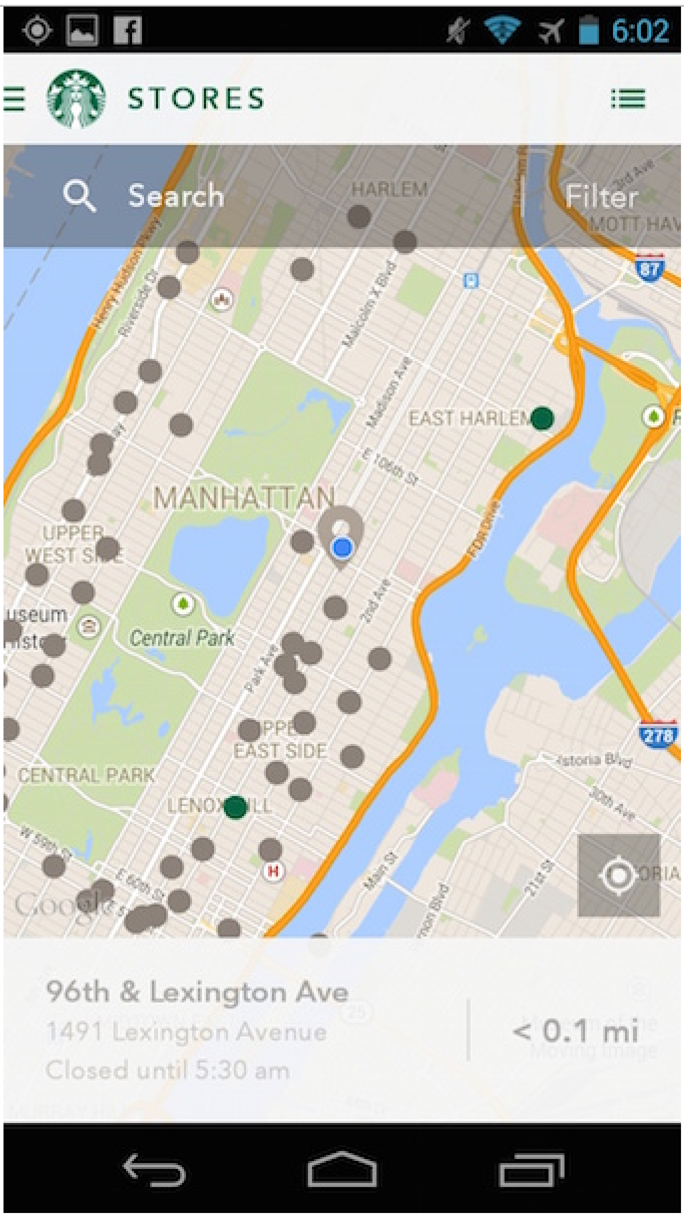
Paper- stories from Facebook

Screen 7 - Aldo on iOS



Aldo on iOS

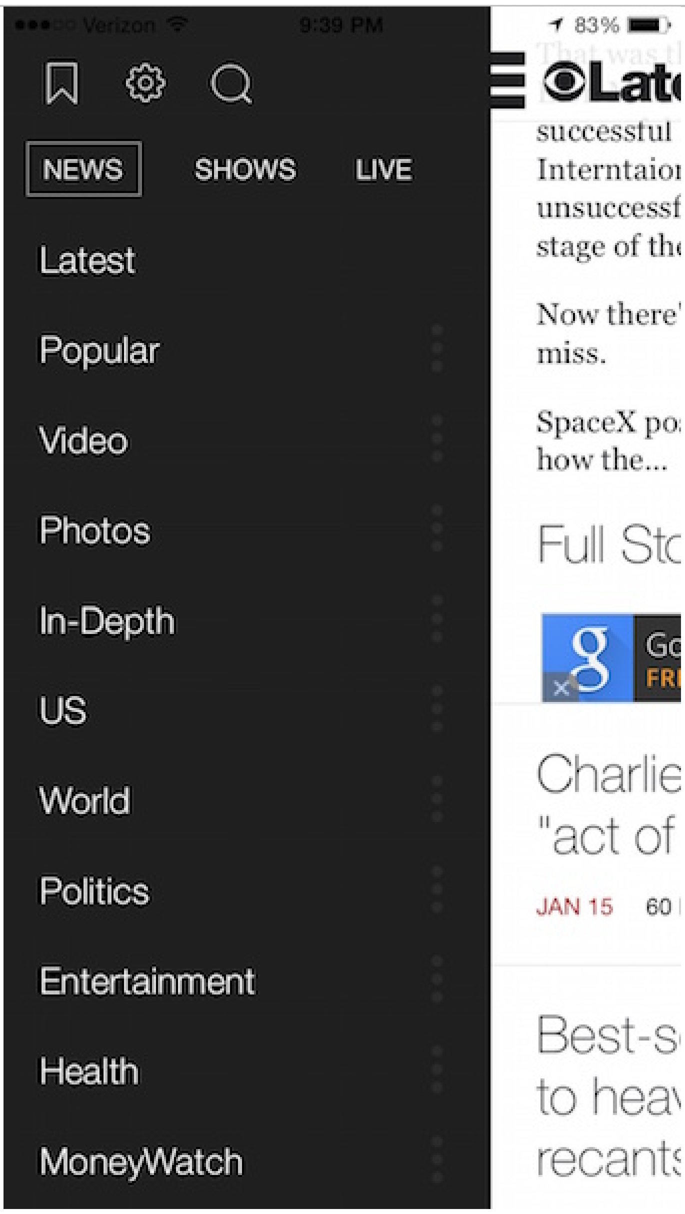
Screen 8 - Starbucks on Android



96th & Lexington Ave
1491 Lexington Avenue
Closed until 5:30 am
< 0.1 mi

Starbucks on Android

Screen 9 - CBS News on iOS



That was tl
successful
Interntaion
unsuccessf
stage of th

Now there'
miss.

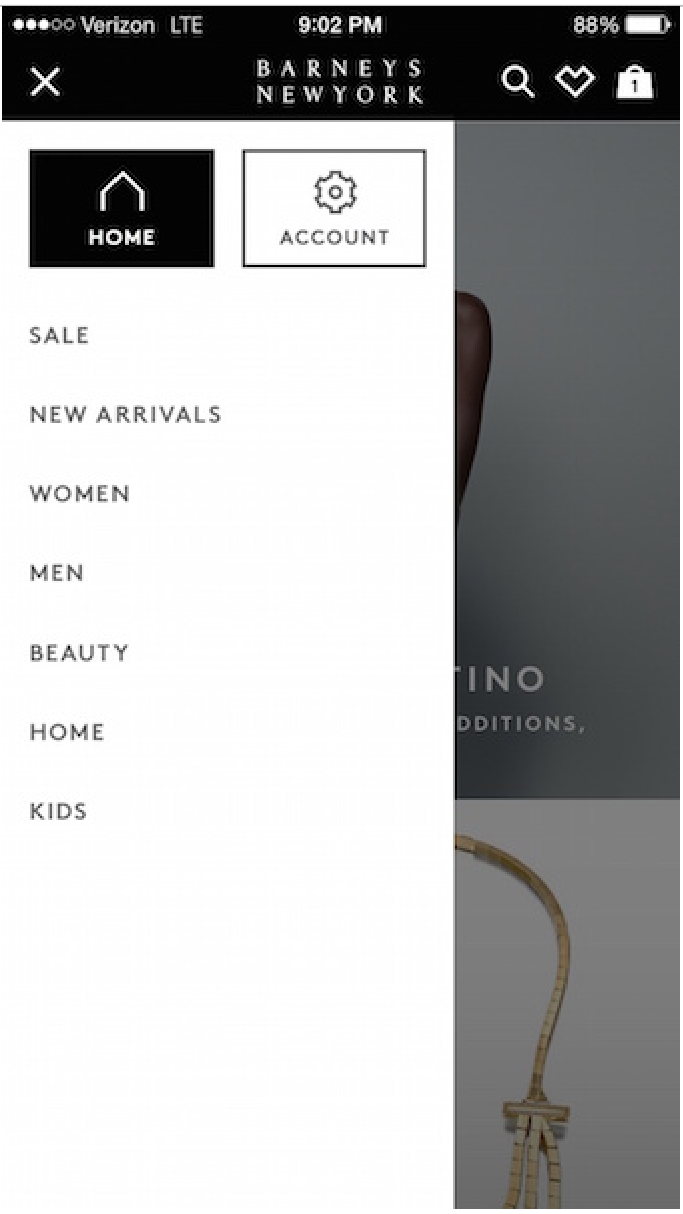
SpaceX po
how the...

Charlie
"act of

Best-s
to hea
recants

CBS News on iOS

Screen 10 - Barneys New York on iOS



Barneys New York on iOS

References

- Cordova API docs
  - *config.xml*
  - *Globalization*
  - *Hooks*
  - *Merges*
  - *Network Information*
  - *Whitelisting*
- OnsenUI
  - *JavaScript Reference*