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

Fall Semester 2017 - Week 9

Dr Nick Hayward

notes app - intro

Some initial considerations for IndexedDB

- connecting
- event listeners
- creating a new DB
- adding object stores
- transactions
- adding data, reading data...
- index and querying...

Cordova app - NoteTaker - v1

notes app - basic requirements

Start building initial **Trip Notes** app,

- Cordova base app created
- add support for required platforms
  - *Android & Browser (for testing...)*
- add any initial plugins
- set icon and splashscreens in `config.xml`

Then build out initial app,

- add home screen
  - *initial design and layout*
  - *initial widgets and elements*
  - *add some basic styling*
- add IndexedDB support
  - *create base **object stores***
  - *load some notes*
- add some more UI options
  - *change view of notes - grid, list...*
  - *sort and filter notes*
- view single note

Then, move on to v2...

Cordova app - NoteTaker - v1

notes app - icon and splashscreen

- add the splashscreen plugin,

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

- then update `config.xml` file
  - *add support for required splashscreens and icons*
- set a value for the splashscreen timeout
  - *controls default AutoHide for splashscreen*

```
<preference name="SplashScreenDelay" value="3000" />
```

- option can be overridden
  - *either programmatically in JS*
  - *or in the `config.xml` file, e.g.*

```
<preference name="AutoHideSplashScreen" value="false" />
```

- default value set to `value="true"`
  - *hides splashscreen at specified value for delay*
- if `AutoHideSplashScreen` set to false
  - *needs to be hidden programmatically in the app's JS*

notes app - `config.xml` - splashscreens and icons

Add splashscreens and icons to the `config.xml` file

```
<platform name="android">
    <icon density="ldpi" src="resources/android/icon/drawable-ldpi-icon.png" />
    <icon density="mdpi" src="resources/android/icon/drawable-mdpi-icon.png" />
    <icon density="hdpi" src="resources/android/icon/drawable-hdpi-icon.png" />
    <icon density="xhdpi" src="resources/android/icon/drawable-xhdpi-icon.png" />
    <icon density="xxhdpi" src="resources/android/icon/drawable-xxhdpi-icon.png" />
    <icon density="xxxhdpi" src="resources/android/icon/drawable-xxxhdpi-icon.png" />
    <splash density="land-ldpi" src="resources/android/splash/drawable-land-ldpi-screen.png" />
    <splash density="land-mdpi" src="resources/android/splash/drawable-land-mdpi-screen.png" />
    <splash density="land-hdpi" src="resources/android/splash/drawable-land-hdpi-screen.png" />
    <splash density="land-xhdpi" src="resources/android/splash/drawable-land-xhdpi-screen.png" />
    <splash density="land-xxhdpi" src="resources/android/splash/drawable-land-xxhdpi-screen.png" />
    <splash density="land-xxxhdpi" src="resources/android/splash/drawable-land-xxxhdpi-screen.png" />
    <splash density="port-ldpi" src="resources/android/splash/drawable-port-ldpi-screen.png" />
    <splash density="port-mdpi" src="resources/android/splash/drawable-port-mdpi-screen.png" />
    <splash density="port-hdpi" src="resources/android/splash/drawable-port-hdpi-screen.png" />
    <splash density="port-xhdpi" src="resources/android/splash/drawable-port-xhdpi-screen.png" />
    <splash density="port-xxhdpi" src="resources/android/splash/drawable-port-xxhdpi-screen.png" />
    <splash density="port-xxxhdpi" src="resources/android/splash/drawable-port-xxxhdpi-screen.png" />
</platform>
<preference name="SplashScreenDelay" value="3000" />
```

n.b. *we'll initially set `SplashScreenDelay` in the `config.xml` file...*

Cordova app - NoteTaker - v1

notes app - initial home screen

Initial requirements for app's home screen

Places

- header & navbar
- title
- icons for *create note*, *menu*...
- main content
- heading
- grid for notes
- footer

Functionality

- view all notes
- single note with title, snippet, &c.
- no. of notes
- switch layout of notes
- grid, list, &c.
- filter and sort notes
- ...

Cordova app - NoteTaker - v1 - jQuery Mobile

notes app - index.html - part 1

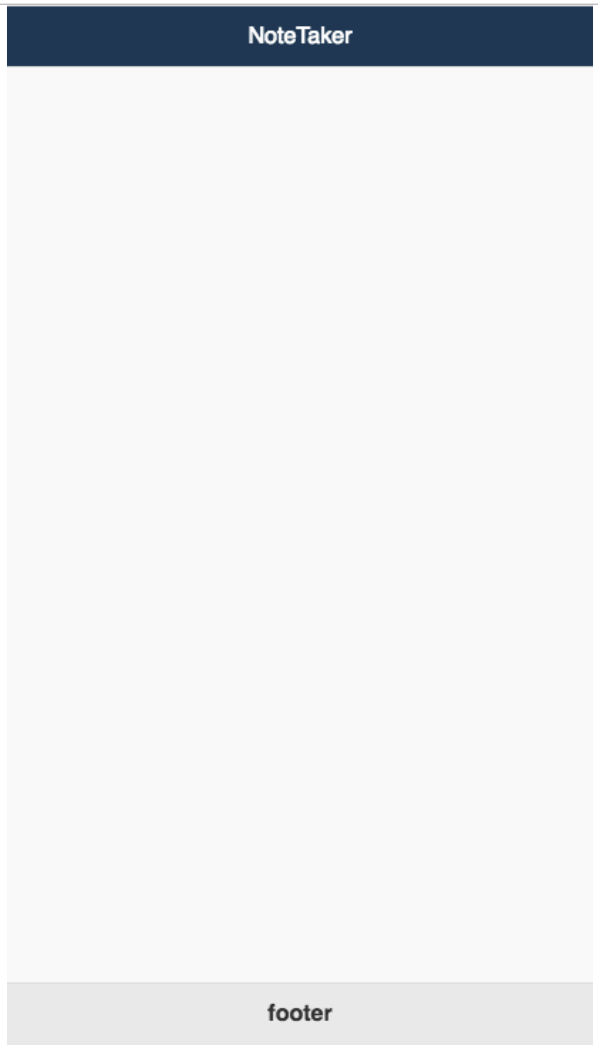
- update app's `index.html` page using jQuery Mobile structure

```
<body>
    <!-- app pages -->
    <!-- page1 - home screen -->
    <div data-role="page" id="home">
        <div data-role="header">
            <h3>NoteTaker</h3>
        </div><!-- /header -->
        <div role="main" class="ui-content">
            ...
        </div><!-- /content -->
        <div data-role="footer" data-position="fixed" class="page-footer" >
            <h5>footer</h5>
        </div><!-- /footer -->
    </div><!-- /page1 - home screen -->
</body>
```

- modify the initial CSS for the app, and add specifics for jQuery Mobile

```
/* remove default all uppercase...*/
body {
    text-transform: none;
}
/* customise page header */
.ui-page .ui-header {
    background-color: #1a3852;
}
/* customise header title */
.ui-header .ui-title {
    font-weight: normal;
    text-shadow: none;
    color: #fff;
}
```

Image - NoteTaker - Home Screen 1 - jQuery Mobile



NoteTaker app - initial home screen 1

Cordova app - NoteTaker - v1 - OnsenUI setup

- now start to introduce alternative UI frameworks for developing cross-platform apps
  - *first option is **OnsenUI***
- supports development with
  - *JavaScript*
  - *Angular 1 & 2*
  - *React*
- setup initial Cordova project, add platforms, plugins &c.
- then add OnsenUI to newly created project
  - *use either NPM or Bower*
- install Bower using the following terminal command

```
npm install -g bower
```

- use Bower to install UI components and dependencies for building our OnsenUI projects

```
cd www
bower install onsenui
```

- `bower_components` directory created in the project's `www` directory

Cordova app - NoteTaker - v1 - OnsenUI

add OnsenUI files to project

- we need to add OnsenUI to our project
  - *add framework's CSS and JS*

```
...
<head>
...
<!-- setup css -->
<link rel="stylesheet" type="text/css" href="css/index.css">
<link rel="stylesheet" type="text/css" href="bower_components/onsenui/css/onsenui.css">
<link rel="stylesheet" type="text/css" href="bower_components/onsenui/css/onsen-css-components.css">
<link rel="stylesheet" type="text/css" href="css/style.css">
<!-- setup js -->
<script type="text/javascript" src="cordova.js"></script>
<script type="text/javascript" src="bower_compoents/onsenui/js/onsenui.js"></script>
<script type="text/javascript" src="js/app.js"></script>
<title>NoteTaker</title>
</head>
...
```

- adding required OnsenUI framework CSS and JavaScript
  - *allow us to use specific OnsenUI elements, structures, methods...*
- update these files to support custom themes and styles
  - *create using the* ***OnsenUI Theme Roller***
  - *add app specific styles to* `css/style.css`

Cordova app - NoteTaker - v1 - OnsenUI

OnsenUI concepts - `ons` object, elements, page content...

- `ons` object is exposed as an integral part of working with OnsenUI
  - *part of the core library*
  - *use with available exposed methods - e.g. with tabbar, page, various utilities...*
- OnsenUI specific elements are all custom
  - *defined with tag prefix of `<ons-`*
  - *elements still include attribute and class patterns*
  - *provide properties and events as expected for standard HTML*
- still traverse an OnsenUI created DOM as expected
- OnsenUI includes many UI components
  - *e.g. `<ons-navigator>` component provides management of page stack and app navigation*

```
<ons-navigator id="navigator" page="page1.html"></ons-navigator>
```

- also add many types of pre-built OnsenUI components, e.g.
  - *buttons, dialogs, notifications*
  - *toolbars, pages, splitters, tabs*
  - *forms, lists...*
- helper properties and functions such as *infinite scroll*
  - *provide pre-built ways to manage content, rendering, layout, and general development...*

Cordova app - NoteTaker - v1 - OnsenUI

notes app - index.html - part 1

- add a page using the <ons-page> element
  - *becomes initial container for a single page*
  - *root element for other page elements*
- add a toolbar to the top or foot of a page
  - *using the <ons-toolbar> element*

```
<!-- page root -->
<ons-page>
  <!-- page toolbar -->
  <ons-toolbar>
  ...
  </ons-toolbar>
</ons-page>
```

- update our toolbar with a menu icon, title, and search icon

```
<!-- page toolbar -->
<ons-toolbar>
    <div class="left">
    <ons-toolbar-button>
      <!-- hamburger menu -->
      <ons-icon icon="md-menu"></ons-icon>
    </ons-toolbar-button>
  </div>
  <div class="center">NoteTaker</div>
  <div class="right">
    <ons-toolbar-button>
      <!-- search option -->
      <ons-icon icon="md-search"></ons-icon>
    </ons-toolbar-button>
  </div>
</ons-toolbar>
```

Image - NoteTaker - add initial navbar - OnsenUI

NoteTaker app - add initial navbar

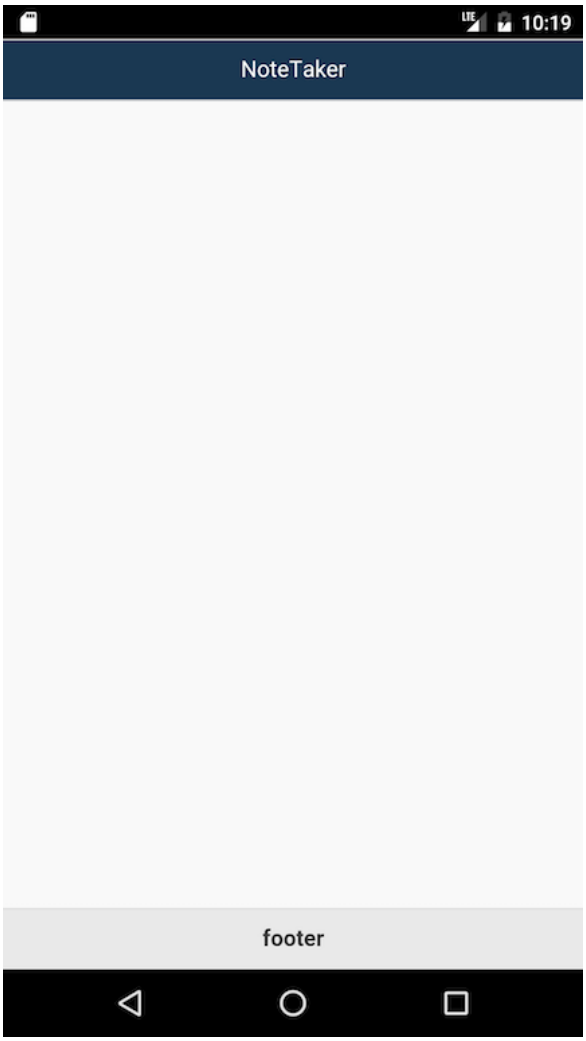
Cordova app - NoteTaker - v1 - OnsenUI

notes app - index.html - part 2

initial homepage,

```
<ons-page id="home">
  <!-- page toolbar -->
  <ons-toolbar>
    <div class="left">
      <ons-toolbar-button>
        <!-- hamburger menu -->
        <ons-icon icon="md-menu"></ons-icon>
      </ons-toolbar-button>
    </div>
    <div class="center">NoteTaker</div>
    <div class="right">
      <ons-toolbar-button>
        <!-- search option -->
        <ons-icon icon="md-search"></ons-icon>
      </ons-toolbar-button>
    </div>
  </ons-toolbar>
  <!-- home page content -->
  <h4>notes</h4>
  ...
  <ons-button id="push-button">Create</ons-button>
</ons-page>
```

Image - NoteTaker - Statusbar - default

jQuery Mobile	OnsenUI
	

Cordova app - NoteTaker - v1

notes app - statusbar

- add Cordova's `statusbar` plugin
  - *helps customise statusbar at the top of the UI*

```
cordova plugin add cordova-plugin-statusbar
```

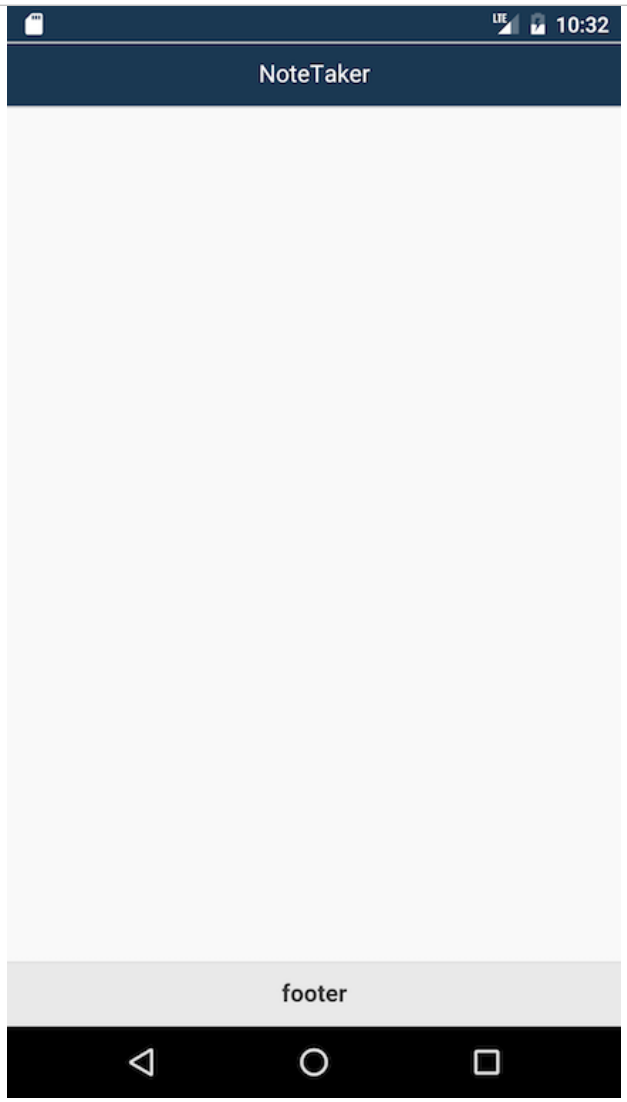
- update app's `config.xml` file to modify background colour

```
<preference name="StatusBarBackgroundColor" value="#1a3852" />
```

- status bar now matches aesthetics of app's colour scheme
- many other options detailed in the Cordova API

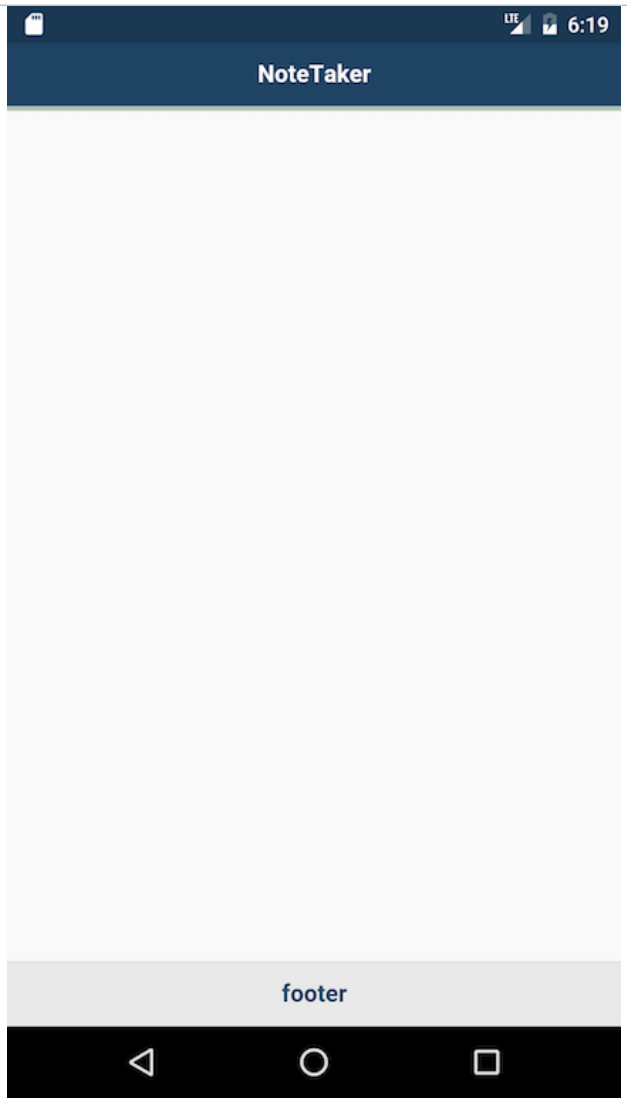
n.b. *Cordova Plugin Statusbar*

Image - NoteTaker - Statusbar - custom



NoteTaker app - custom statusbar

Image - NoteTaker - Statusbar - custom



NoteTaker app - custom statusbar

Cordova app - NoteTaker - v1 - jQuery Mobile

notes app - index.html - part 2

- start to add content to home screen
  - *including create button for new note, grid layout for notes, second page for create note form...*

```
<!-- header -->
<div data-role="header" class="ui-nodisc-icon">
    <h3>NoteTaker</h3>
    <a href="#create" data-transition="slideup"
    class="ui-btn ui-icon-plus ui-btn-icon-notext ui-btn-right">create</a>
</div><!-- /header -->
```

- updates header - adds *plus* icon for *create note* option
- need to match app's aesthetics
  - *need to modify svg properties for underlying icon*

```
/* custom svg for button - plus - custom fill colour = 1f4463*/
.ui-icon-plus:after {
    background-image: url("...polygon%20fill%3D%22%231f4463...");
}
```

- modify *polygon fill* to fit our app's colour scheme

```
polygon%20fill%3D%22%231f4463
```

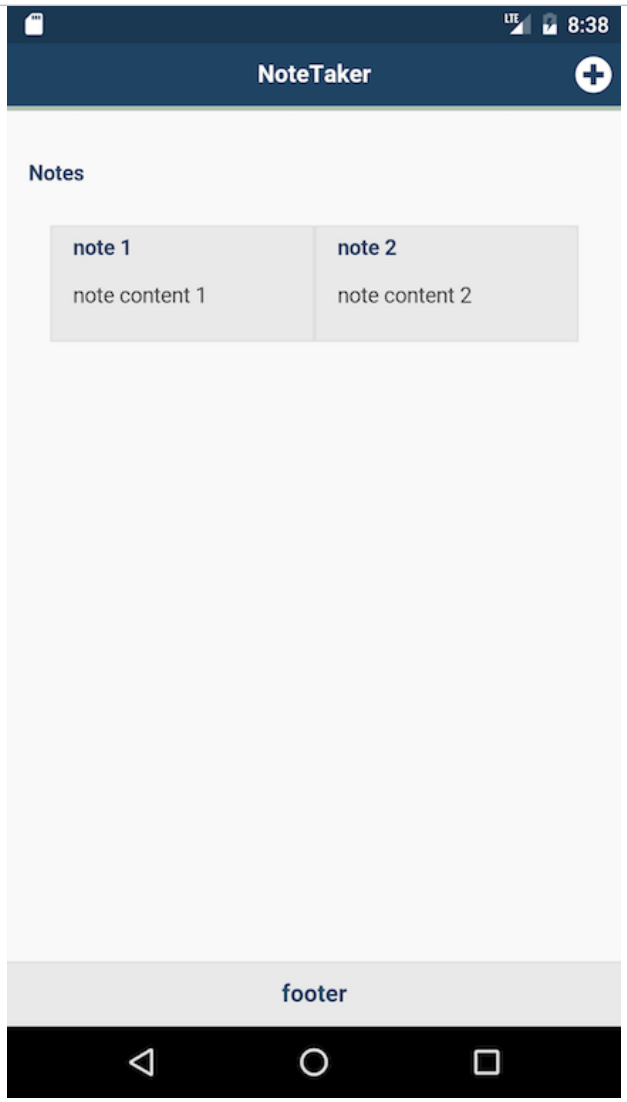
Cordova app - NoteTaker - v1 - jQuery Mobile

notes app - index.html - part 3

- notes rendered by default using a grid pattern
  - *contains note's title and snippet of content*
- grid uses the following pattern
  - *abstract in JS as we read notes from specified data store*

```
<div role="main" class="ui-content">
    <h4>Notes</h4>
    <!-- output available notes -->
    <div id="notes" class="ui-body">
        <!-- grid layout for notes -->
        <div class="ui-grid-a">
            <!-- left column -->
            <div class="ui-block-a">
                <div class="ui-bar ui-bar-a">
                    <h5>note 1</h5>
                    <p>note content 1</p>
                </div>
            </div>
            <!-- right column -->
            <div class="ui-block-b">
                <div class="ui-bar ui-bar-a">
                    <h5>note 2</h5>
                    <p>note content 2</p>
                </div>
            </div>
        </div>
    </div>
</div><!-- /content -->
```

Image - NoteTaker - update `index.html` - jQuery Mobile



NoteTaker app - update `index.html`

Cordova app - NoteTaker - v1 - OnsenUI

notes app - index.html - part 3

- add a second page to allow a user to create their notes
- follow the same pattern as the home page

```
<ons-page id="create">
  <ons-toolbar>
    <div class="left"><ons-back-button>Back</ons-back-button></div>
    <div class="center"></div>
  </ons-toolbar>
  <!-- create note page -->
  <h4>create note</h4>
  ...
</ons-page>
```

- update toolbar relative to page requirements

notes app - page lifecycle

<ons-page> element provides the following events at different points during the lifecyle of a page

- `init`
  - *fired after page is added to DOM*
- `destroy`
  - *fired prior to page being removed from the DOM, and just before the page is destroyed*
- `show`
  - *fired every time <ons-page> comes into the view*
- `hide`
  - *fired every time <ons-page> disappears from the view*

Cordova app - NoteTaker - v1 - OnsenUI

notes app - JS and initial listeners

- using OnsenUI and Cordova we need to consider specific events as an app starts...
- DOM loads - attach listener for Cordova's `deviceready`

```
//add listener to DOM - check deviceready event...continue with app logic
document.addEventListener('deviceready', onDeviceReady, false);
```

- JS checks require DOM has loaded
  - *otherwise we can't attach listeners for such events*
- `deviceready` event crucial to app loading
- if `deviceready` event not completed successfully
  - *not able to attach other listeners to the DOM*
- Cordova modifies the default event listener for an app's webview
  - *modifications include handling of special events*
  - *e.g. attaching and removing listeners...*
- as noted in the Cordova JS library,

> *Intercept calls to addEventListener + removeEventListener and handle deviceready, resume, and pause events.*

- `deviceready` event returns successful
  - *next listener needs to check that OnsenUI `init` event has fired*

```
document.addEventListener('init', function(event) {
  //check defined initial page for app...
  if (event.target.matches('#home')) {
     ...
  }
}, false);
```

Cordova app - NoteTaker - v1 - OnsenUI

notes app - notetaker.js

- add the logic for our OnsenUI based app
- start by checking for Cordova's `deviceready` event
- then check the `init` event that OnsenUI provides
  - *use this to check for our home page*

```
...
//cordova - add listener to DOM & check deviceready event
document.addEventListener('deviceready', function(event) {
    //prevent any bound defaults
    event.preventDefault();
    console.log("cordova checked...device ready");

    //call as ons-page added to DOM...
    function onsInit(event) {
        //properties - initial page load
        var page = event.target;
    //check IndexedDB
    //set navigation
        //check for home page
        if (page.matches("#home")) {
        //ons.notification.alert('init checked...homepage ready');
            console.log("home page is now attached to the DOM...");
      } else {
            console.log("away from home page...");
        }
    }
    //onsen - init event is fired after ons-page attached to DOM...
    document.addEventListener('init', onsInit, false);
}, false);
...
```

Cordova app - NoteTaker - v1 - OnsenUI

notes app - navigation structure - part 1

- two initial pages for our NoteTaker app - navigation from place to place
  - *update our `index.html` page's structure*
  - *updare app's logic in the `notetaker.js` file*
- provide support for multi-page navigation - three available navigation patterns
  - *add a navigator, tabbar, or splitter component*
  - *each component provides a **frame** for a defined place within our app*
  - *dynamically update the inner content of each frame*
- a **frame** normally contains a `<ons-page>` component
  - *we can also nest multiple navigation components*

Cordova app - NoteTaker - v1 - OnsenUI

notes app - navigation structure - part 2

- add a navigator component
  - *define each individual page within our single page app's `index.html`*
- defined each page for navigation using `<ons-template>` component
- updated structure is as follows

```
<ons-navigator id="navigator" page="home.html"></ons-navigator>
<!-- home page -->
<ons-template id="home.html">
  <ons-page id="home">
   ...
  </ons-page>
</ons-template>
<!-- create note page -->
<ons-template id="create.html">
  <ons-page id="create">
    ...
  </ons-page>
</ons-template>
```

- for a SPA
  - *each `<ons-page>` component's ID attribute replaces a full URL to a separate page*

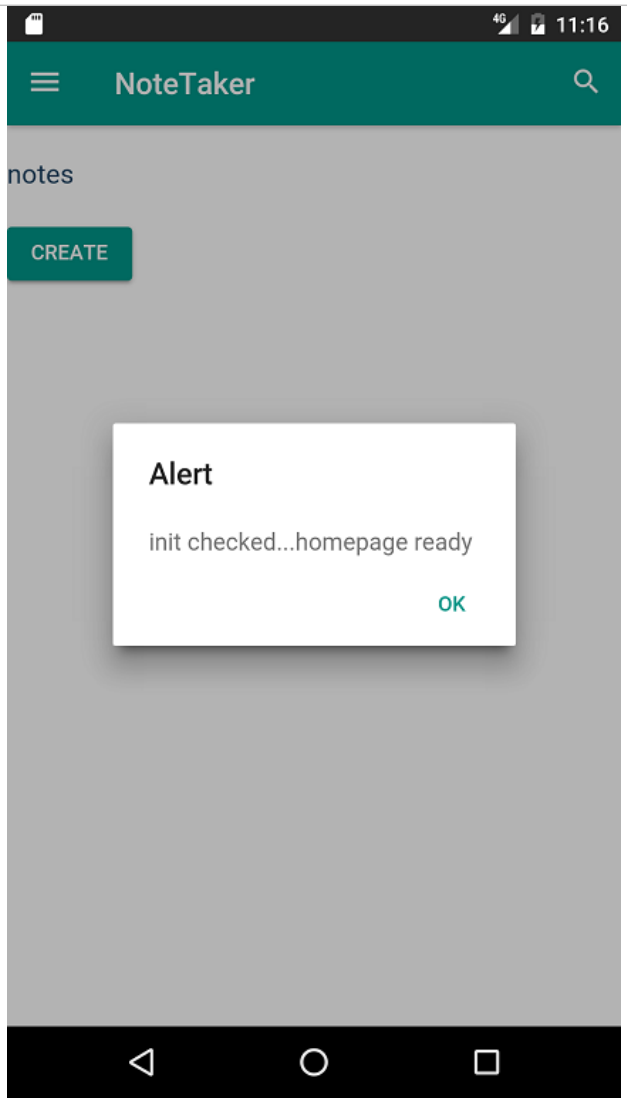
Cordova app - NoteTaker - v1 - OnsenUI

notes app - navigation logic

- update our app's logic to listen for navigation events
- add a custom function `onsNav()`
  - *sets required **stack-based** navigation*
  - *OnsenUI uses to keep track of page push and pop requests within an app*

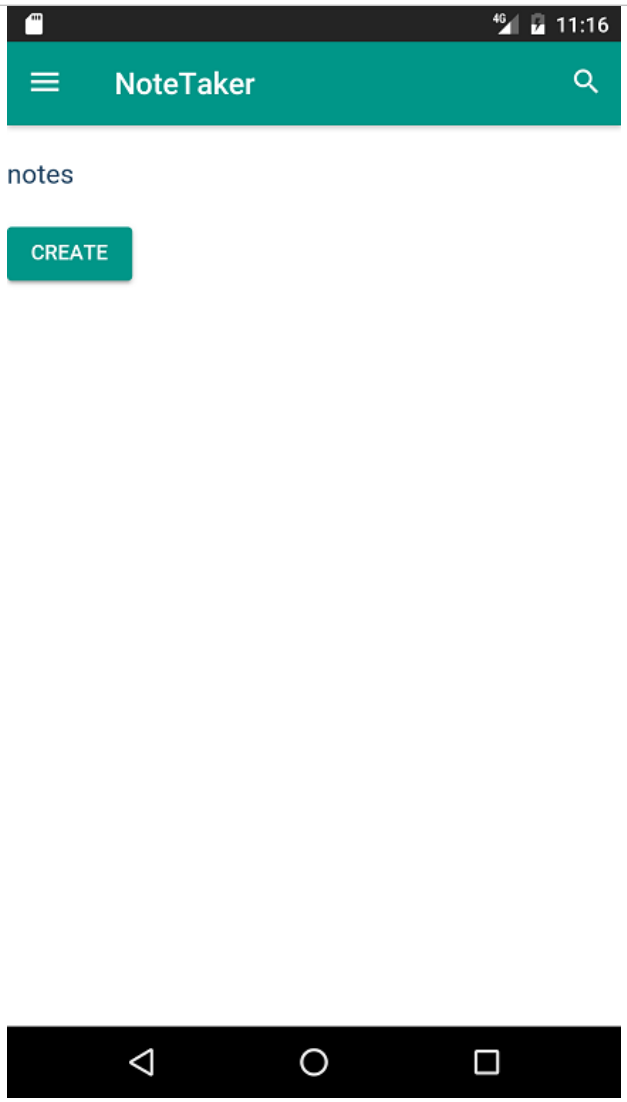
```
//onsen - set stack-based navigation
function onsNav(page) {
  if (page.id === 'home') {
    page.querySelector('#push-button').onclick = function() {
      document.querySelector('#navigator').pushPage('create.html', {data: {title: 'Create Note'}});
    };
  } else if (page.id === 'create') {
    page.querySelector('ons-toolbar .center').innerHTML = page.data.title;
      console.log("page title = "+ page.data.title);
  }
}
```

Image - NoteTaker - check `init` event - OnsenUI



NoteTaker app - check init event

Image - NoteTaker - load home page - OnsenUI



NoteTaker app - load home page

Image - NoteTaker - add navigation - OnsenUI

NoteTaker app - add navigation

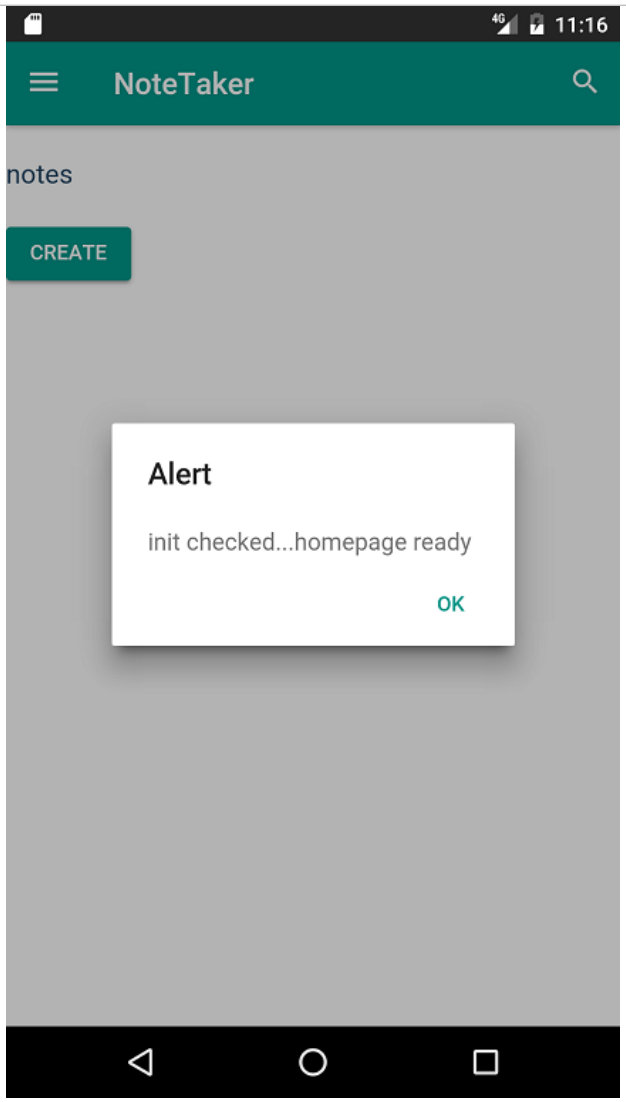
notes app - recap

Latest app features and updates,

- home page and create note page
- initial navigation stack
- statusbar customisation and titles
- splashscreens and icon
- initial page elements
- checked loading of
  - `deviceready` *for Cordova*
  - `init` *for OnsenUI ons-page component*

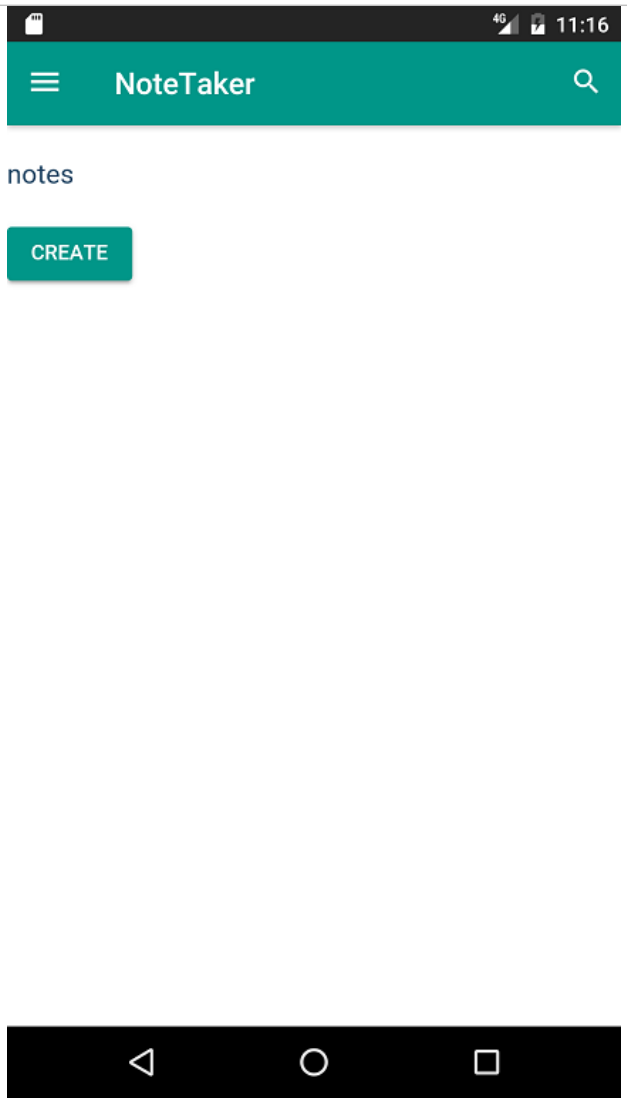
and a few updates to the general aesthetics...

Image - NoteTaker - check `init` event - OnsenUI



NoteTaker app - check init event

Image - NoteTaker - load home page - OnsenUI



NoteTaker app - load home page

Image - NoteTaker - add navigation - OnsenUI

NoteTaker app - add navigation

Cordova app - NoteTaker - v1 - OnsenUI

notes app - update UI

- start to modify our initial UI for our OnsenUI based app
- add a standard Material Design icon for creating a note
- use with our existing navigation stack
- standard floating action button pattern
  - *defined in the Material Design specification*

```
<ons-fab position="bottom right">
  <ons-icon id="create-note" icon="md-plus"></ons-icon>
</ons-fab>
```

- Material Design specification - Floating Action Button

Cordova app - NoteTaker - v1 - OnsenUI

notes app - update UI - grid layout

- need to use a grid layout for our initial notes
- consult the Material Design guidelines for a **grid list**
  - *Material Design Guidelines - grid list*
- OnsenUI provides components for rows and columns, e.g.

```
<ons-row>
  <ons-col>
    <div class="note-card">
      <h5>note 1</h5>
    </div>
  </ons-col>
  <ons-col>
    <div class="note-card">
      <h5>note 2</h5>
    </div>
  </ons-col>
</ons-row>
```

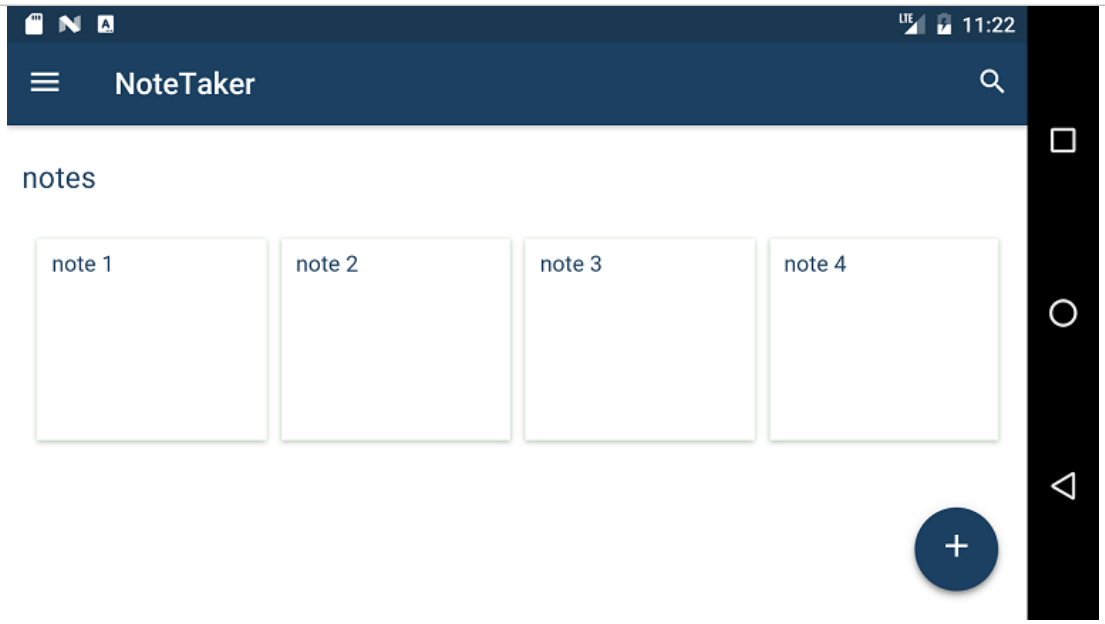
- also add a custom class to recreate some cards for our notes, e.g.

```
.note-card {
box-shadow: 0 1px 4px #b1c4b1;
background-color: #ffffff;
}
```

Image - NoteTaker - grid layout portrait - OnsenUI

NoteTaker app - grid layout - portrait

Image - NoteTaker - grid layout landscape - OnsenUI



NoteTaker app - grid layout - landscape

Cordova app - NoteTaker - v1 - OnsenUI

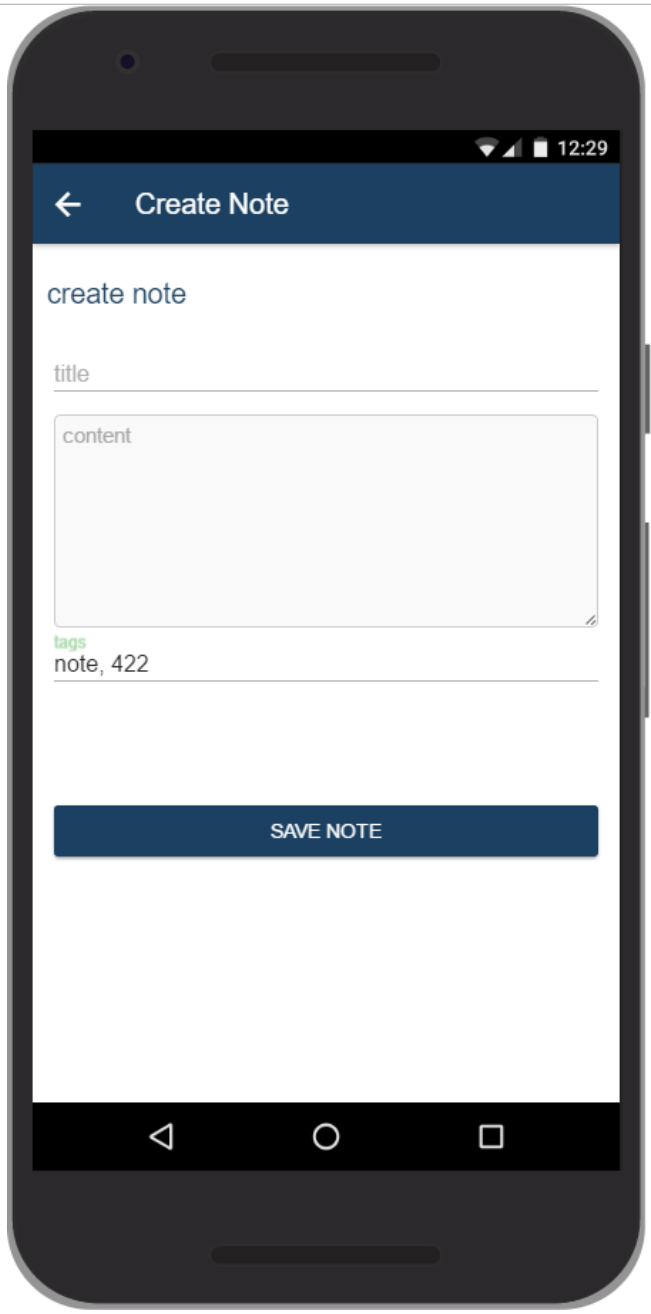
notes app - create note page

- next initial update
  - *add options for a user to create their notes on the **create note** page*
- need to add a form with various fields for a note
- `<ons-input>` component for input fields
  - *component supports many different common form elements*
  - *checkbox, radio button, password field...*
- need the following minimum elements and options for our **create note** page
  - *title, content, tags*

```
<section style="padding: 10px 0 10px 0">
  <ons-input type="text" placeholder="title" modifier="material"
  style="display: block; width: 100%"></ons-input>
</section>
<section style="padding: 5px 0 5px 0">
  <textarea class="textarea" placeholder="content" modifier="material"
  style="width: 100%; height: 150px; resize: vertical;"></textarea>
</section>
<section style="padding: 10px 0 10px 0">
  <ons-input type="text" placeholder="tags" modifier="material"
  style="width: 100%; height: 100px; resize: vertical;"></ons-input>
</section>
<section>
  <ons-button modifier="large">save note</ons-button>
</section>
```

- mixture of standard HTML5 elements and OnsenUI components
  - *desired layout and rendering for our create note page*

Image - NoteTaker - create note page - OnsenUI



NoteTaker app - create note page - initial design

Cordova app - NoteTaker - v1 - OnsenUI

notes app - update UI - navigation and splitter structure

- current navigation uses `<ons-navigator>` component
  - *push **create note** page to navigation stack*
- `<ons-splitter>` component
  - *use to create a main menu option*
- component offers different frames that allow us to render varied content
  - *e.g. add our menu with a left splitter - contains menu links, title...*
  - *each link loads requested URL to `<ons-splitter-content>`*
- frames normally contain a `<ons-page>` component
  - *also nest multiple navigation components*
  - *e.g. `<ons-navigator>`*
- basic usage, e.g.

```
<ons-splitter>
  <ons-splitter-side id="menu" side="left" width="220px" collapse swipeable>
    <ons-page>
      <ons-list>
        ...
      </ons-list>
    </ons-page>
  </ons-splitter-side>
  <ons-splitter-content id="content" page="home.html"></ons-splitter-content>
</ons-splitter>
```

Cordova app - NoteTaker - v1 - OnsenUI

notes app - update UI - navigation and splitter structure

- combine the navigator and splitter component
  - *need to consider how they will complement each other*
  - *ensure navigation stack works correctly*
- structure of our HTML needs to be updated
  -

```
<ons-navigator id="navigator" page="splitter.html"></ons-navigator>
<ons-template id="splitter.html">
  <ons-splitter>
    <ons-splitter-side id="menu" side="left" width="220px" collapse swipeable>
      <ons-page id="menu.html">
        <ons-list>
          <ons-list-item url="home.html" class="menu-link" tappable>home</ons-list-item>
          <ons-list-item url="about.html" class="menu-link" tappable>about</ons-list-item>
        </ons-list>
      </ons-page>
    </ons-splitter-side>
    <ons-splitter-content id="content" page="home.html"></ons-splitter-content>
  </ons-splitter>
</ons-template>
```

Cordova app - NoteTaker - v1 - OnsenUI

notes app - update UI - navigation and splitter promises

- slightly different from the prescribed pattern in the OnsenUI docs
- after testing an initial pattern
  - *navigator component as a parent container to the splitter component*
- initially errors reported relative to blocked, existing promises
- splitter and its associated animation was in conflict
  - *with subsequent calls to the menu itself*
  - *and the navigator component*
- forum answer was an initial concern, not a satisfactory resolution
  - *Onsen Community*
- this issue led to the previous design and updated JS logic
- no longer blocks the required promises...

Cordova app - NoteTaker - v1 - OnsenUI

notes app - update UI - navigation and splitter logic

- relative to our menu option
  - *add this check to force the logic*
  - *checks target page before loading the menu itself*
- if not, execution of JS logic will return 'null' for requested menu open selector. e.g.

```
if (event.target.id === 'home') {
  //get menu icon - query selector OK due to one per ons page
    var menuOpen = document.querySelector('.menu-open');
    //check menu open is stored...
    if (menuOpen) {
      console.log("menu open stored...");
    }
}
```

- checking that we can actually now use the menu open selector
  - *toggle state of the menu*
  - *then add an event listener for the main menu*
  - *allows us to open the menu on any applicable ons page*

```
//add event listener for main menu
menuOpen.addEventListener('click', function(event) {
  event.preventDefault();
  //open main menu for current page
  menu.open();
}, false);
```

Cordova app - NoteTaker - v1 - OnsenUI

notes app - update UI - navigation and splitter logic

- need to handle multiple possible links in the menu itself
  - *ensure requested page is loaded in the splitter content*

```
if (event.target.id === 'menu.html') {
  console.log("menu target...");
  //es6 Array.prototype.forEach iteration...
  Array.from(menuLink).forEach(link => {
    link.addEventListener('click', function(event) {
      event.preventDefault();
      var url = this.getAttribute('url');
      console.log("menu link = "+ url);
      content.load(url)
      .then(menu.close.bind(menu));
    }, false);
  });
}
```

- updated navigator and splitter component structure helps with overall logic
- check and add a listener for each menu item
  - *as and when the menu is actually loaded in the app*
  - *resolves situation of locked promises for splitter component...*
- allows us to correctly select our menu, and menu items
- also select **create note** option as well on a given page
  - *set navigation stack for the **create note** option by checking against given page event*

```
if (event.target.id === 'home') {
    //set navigation
  onsNav(event.target);
}
```

Cordova app - NoteTaker - v1 - OnsenUI

notes app - update UI - splitter, navigation, and backbutton

- using the `<ons-splitter>` and `<ons-navigator>` components
  - *helps create the correct structure for our app*
- need to consider interaction with hardware backbutton on Android
- Android device default usage pattern
  - *default behaviour for hardware backbutton = close the app*
  - *user may reopen app from recent items using the overview button*
- Android behaviour pattern replicated by Cordova
  - *fires event to handle hardware button within an app*
  - *part of the default `cordova.js` file*
- OnsenUI also set handlers for this hardware button for given UI components
  - *`Dialogs` - close a cancelable dialog*
  - *`Navigator` - if page stack not empty, pops a page from navigation stack*
  - *`Splitter` - close the menu if currently open*

Cordova app - NoteTaker - v1 - OnsenUI

notes app - update UI - splitter, navigation, and backbutton

- menu system based upon the splitter component
  - *careful how we handle this hardware back button*
  - *default action is to simply exit an app*
- can update or modify this behaviour to create explicit action
  - *offer feedback to users before an **exit** is executed*

```
// initially disable hardware backbutton on Android
ons.disableDeviceBackButtonHandler();

// set custom backbutton handler
ons.setDefaultDeviceBackButtonListener(function(event) {
  ons.notification.confirm('Exit app?') // check with user
  .then(function(index) {
    if (index === 1) { // 'ok' button
      navigator.app.exitApp(); // default behaviour - exit app
    }
  });
});
```

- another option might simply be to maintain an in-app tracker
  - *track pages pushed and popped relative to the splitter component*
- still need to be aware of the default, expected behaviour for Android
- should not modify this behaviour too much from default

Cordova app - NoteTaker - v1 - OnsenUI

notes app - update UI - splitter options & structure

- working menu and navigation stack within our initial app
- many different ways to use this splitter option
  - *often informed by an app's page and navigation requirements*
- initially identify the following page and link requirements for our NoteTaker app

Main Menu

```
* home
* media
* notes
* tags
```

navigation stack

```
* create note
* edit note
* tag note
* delete note
* ...
```

Cordova app - NoteTaker - v1 - OnsenUI

notes app - load initial notes

- now need to add our initial notes for the app
- load them as the app starts
- render them on the home screen
- using IndexedDB for app based storage
  - *check and support offline storage for app*
- then save to a cloud based data store
  - *e.g. user requests saving a specific note, notes...*
- add our initial check for IndexedDB support as part of the `deviceready` event

```
//set variable for IndexedDB support
var indexedDBSupport = false;
if("indexedDB" in window) {
  indexedDBSupport = true;
  console.log("IndexedDB supported...");
} else {
  console.log("No support...");
}
```

- create initial variable to store the boolean result
- check variable after `deviceready` event has fired and returned successfully

Cordova app - NoteTaker - v1 - OnsenUI

notes app - load initial notes

- database is local to the browser,
  - *only available to users of the local, native app*
- IndexedDB databases follow familiar pattern of read and write privileges
  - *eg: browser-based storage options, including localstorage*
- create databases with the same name, and then deploy them to different apps
  - *remain domain specific as well*
- first thing we need to do is create an opening to our database

```
var openDB = indexedDB.open("notetaker", 1);
```

- creating a variable for our database connection
  - *specifying the name of the DB and a version*
- open request to the DB is an asynchronous operation

Cordova app - NoteTaker - v1 - OnsenUI

notes app - load initial notes

- create our required DB
  - *check it has persisted during subsequent application loading and usage*
- open a connection to the DB
- checks for three events
  - *upgrade, onsuccess, and any returned errors*
- ready to use the success event
- start to build out our database for our NoteTaker app
  - *add the required initial **object stores***
- also add our required **keypaths**, and a useful index

Cordova app - NoteTaker - v1 - OnsenUI

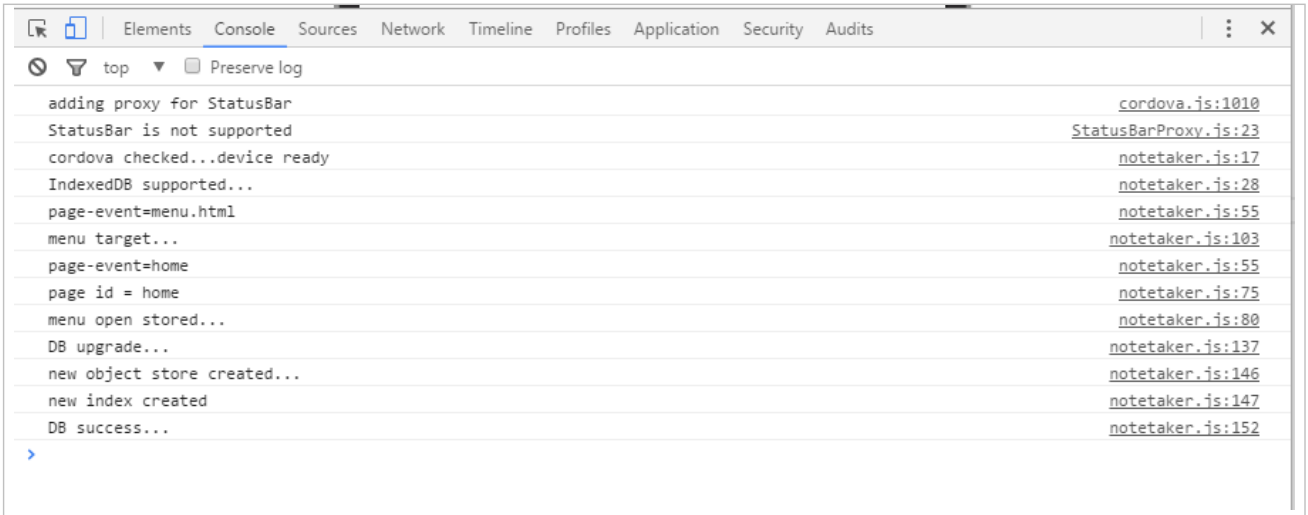
notes app - load initial notes

- update the upgrade event
  - *includes creation of app's required object store*

```
...
openDB.onupgradeneeded = function(e) {
    console.log("DB upgrade...");
    //local var for db upgrade
    var upgradeDB = e.target.result;
    if (!upgradeDB.objectStoreNames.contains("ntos")) {
      upgradeDB.createObjectStore("ntos");
    }
}
...
```

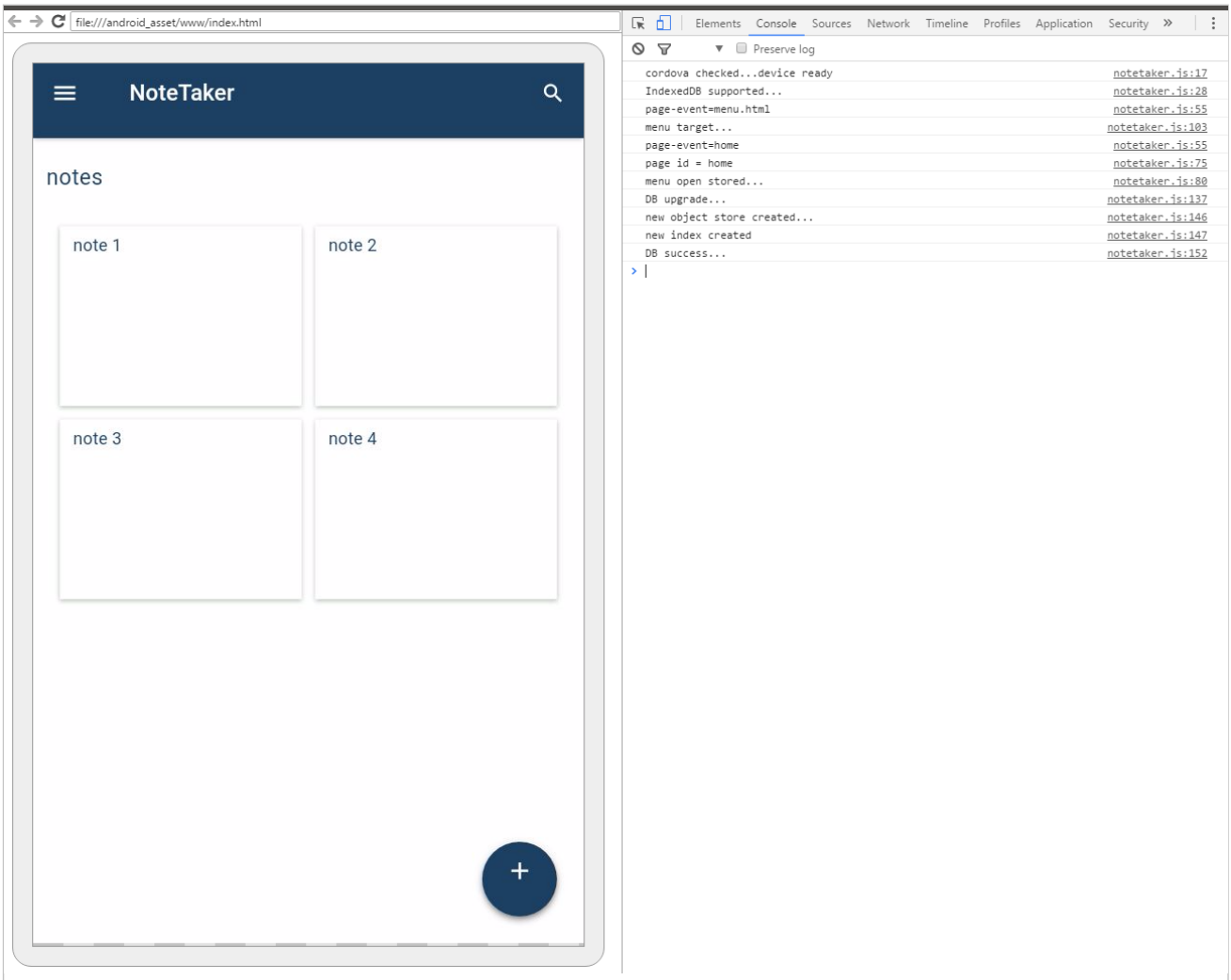
- check a list of existing object stores
- if required object store unavailable we can create our new object store
  - *listen for result from this synchronous method*
- as a user opens our app for the first time
  - *the `upgradeneeded` event is run*
  - *code checks for an existing object store*
  - *if unavailable, create a new one*
  - *then run the `success` handler*

Image - check and load IndexedDB



notetaker app - IndexedDB

Image - check and load IndexedDB



notetaker app - IndexedDB

References

- Cordova API
  - *config.xml*
  - *plugin - Splashscreen*
  - *plugin - statusbar*
- OnsenUI
  - *OnsenUI v2*
  - *JavaScript Reference*
  - *Theme Roller*