Comp 422 - Software Development for Wireless and Mobile Devices

Fall Semester 2016 - Week 12 Notes

Dr Nick Hayward

Contents

Presentation

- Yale's Giuseppe Amatulli on Big Data meets Geo-Computation
- Friday 2nd December 2016
  - *2.45pm to 3.45pm*
- Location
  - *IES-123, Lake Shore Campus*
- continue class after presentation

Final Presentation & Report

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

Final Assessment Outline

- continue to develop your app concept and prototypes using Apache Cordova
- implement a custom Cordova plugin for the following native Mobile OSs
  - *Android*
- produce a working app
  - *as far as possible try to create a fully working app*
  - *explain any parts of the app not working…*
- explain design decisions
  - *outline what you chose and why?*
  - *what else did you consider, and then omit? (again, why?)*
- which concepts could you abstract for easy porting to other platform/OS?
- describe patterns used in design of UI and interaction

Final Assessment Report

- report outline - demo and report

Project Status Report

- what is currently working?
- which data store?
- what is left to add or fix? features, UI elements, interactions...

- who is working on what? logic, design, testing, research...

- team alert
- team bill splitter (study buddies)
- team crunchtime
- team fitness
- team hallpass
- team horatio
- team loyola classifieds
- team loyola group fitness
- team service request

- team walking through naboo

Cordova app - Extras

Other UI Options - Ionic - part 1

- briefly consider option of using Ionic's framework
  - *for developing your UI for Cordova applications*
- Ionic is a HTML framework
  - *designed specifically for development of hybrid applications*
  - *including Cordova mobile applications*
- originally created by a group of developers called **Drifty**
- known to be simple to use and very fast
- Ionic provides
  - *overall UI framework*
  - *accompanying CLI*
- CLI is wrapper for Cordova CLI
- install Ionic using NPM

```
sudo npm install -g ionic
```

- start using Ionic at CLI with `ionic` command

Cordova app - Extras

Other UI Options - Ionic - part 2

- Ionic provides a number of useful starter templates
  - *use and modify for the development of our Cordova applications*
- create a new Ionic project
  - *use the following command at the CLI,*

```
ionic start csteach422 blank
```

- specify the project name
  - *in this example* `csteach422`
- required template for this project
- in this example `blank`
- templates include
  - *Tabs (default) - Demo*
  - *Sidemenu - Demo*
  - *Blank - Demo*
- Ionic CSS Styles - Demo
- Ionic creates a Cordova application
  - *with addition of support and styling for Ionic based UI*

Cordova app - Extras

Other UI Options - Ionic - part 3

- Ionic framework has now used Cordova to build the new project
- also added some Ionic specific components
  - *custom components to help with builds, UI framework updates...*
- Ionic adds platform support for iOS by default
  - *then we can the standard Android support*

```
ionic platform add android
```

- Ionic CLI commands closely match familiar Cordova commands
- a useful command

```
ionic serve
```

- start a local web server and test our project from the working directory
  - *CLI checks preferred server address, eg: `localhost`*
  - *loads project in default browser*

Image - Ionic Starter

Ionic Starter on Android

Image - Ionic Starter

Ionic Starter in the browser

Cordova app - NoteTaker - v1 - OnsenUI

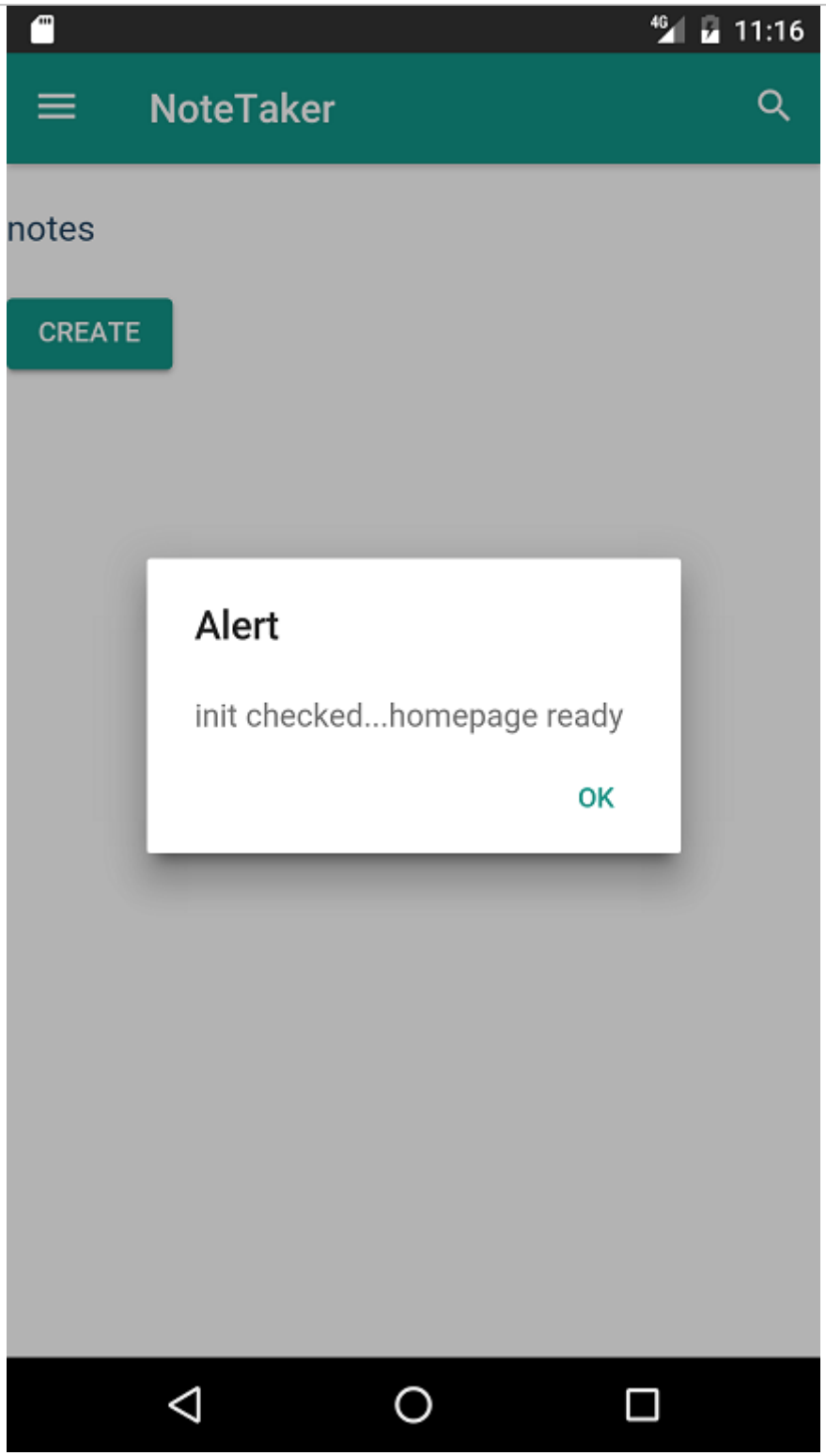
notes app - recap

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*

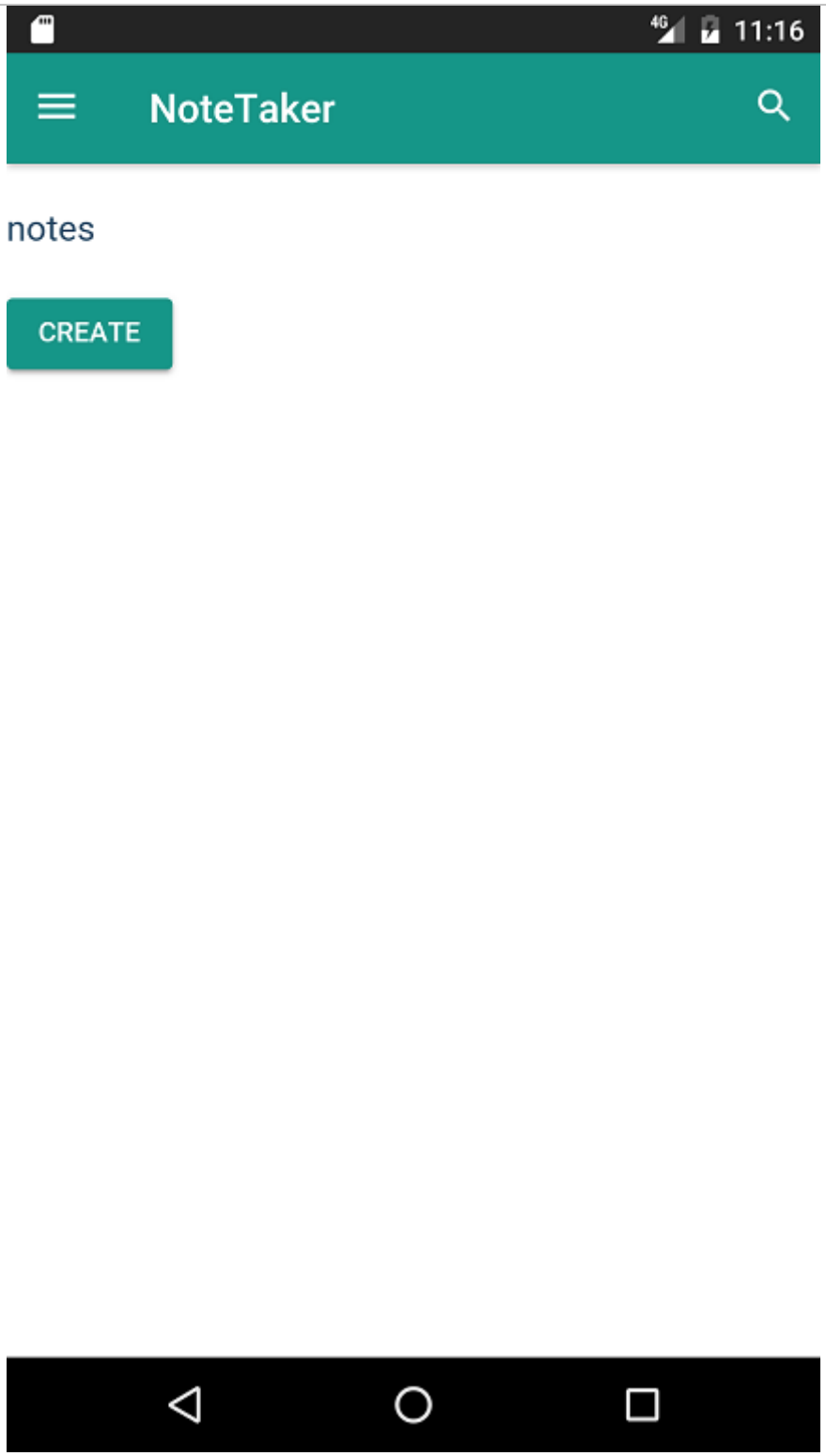
and a few updates to the general aesthetics...

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

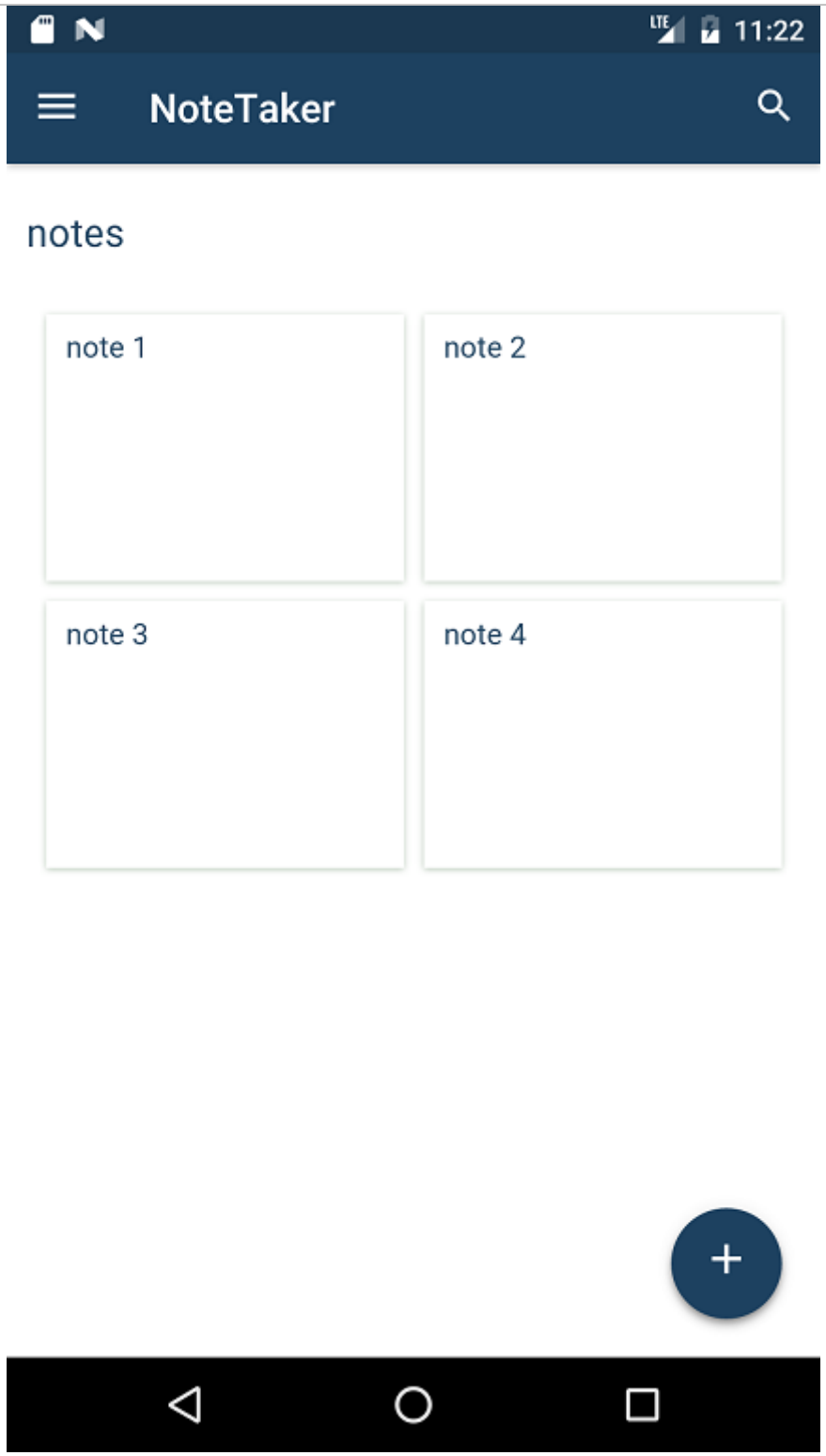
- need to use a grid layout for our initial notes
- consult the Material Design guidelines for a **grid list**
  - *Material Design Guidelines - grid list*
- OnsenUI provides components for rows and columns, e.g.

```
<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.

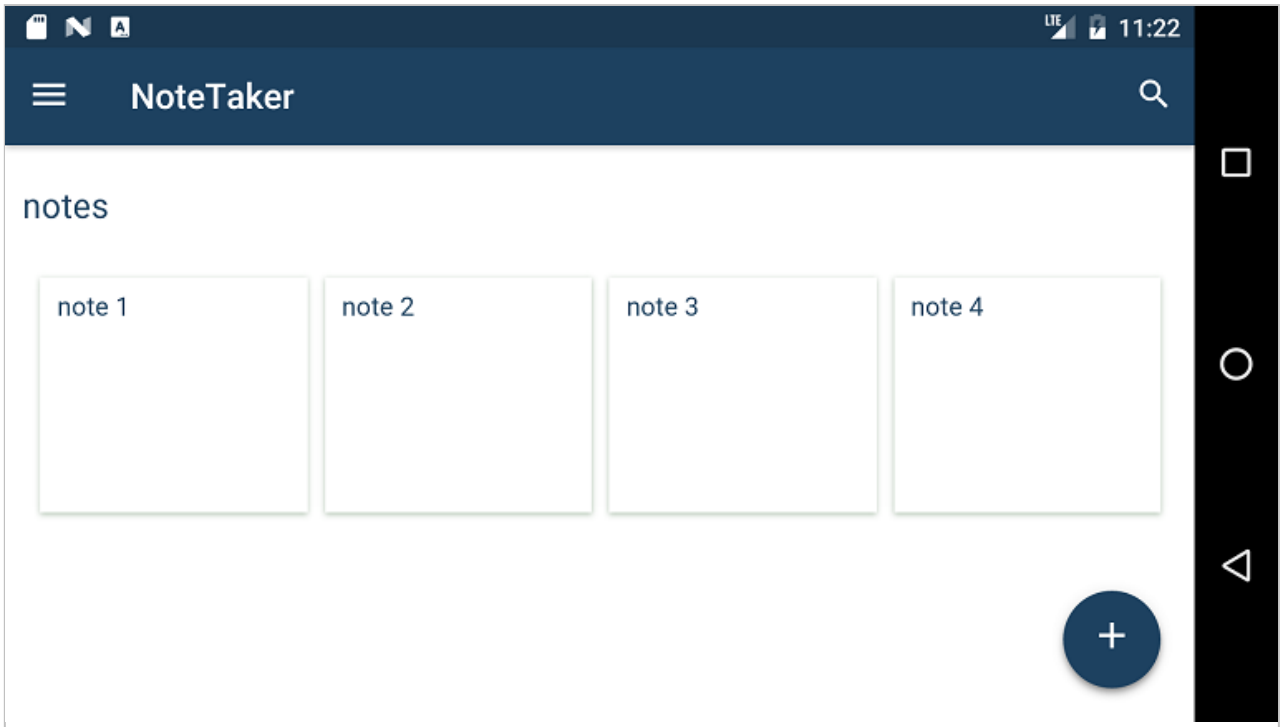
```
.note-card {
box-shadow: 0 1px 4px #b1c4b1;
background-color: #ffffff;
}
```

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

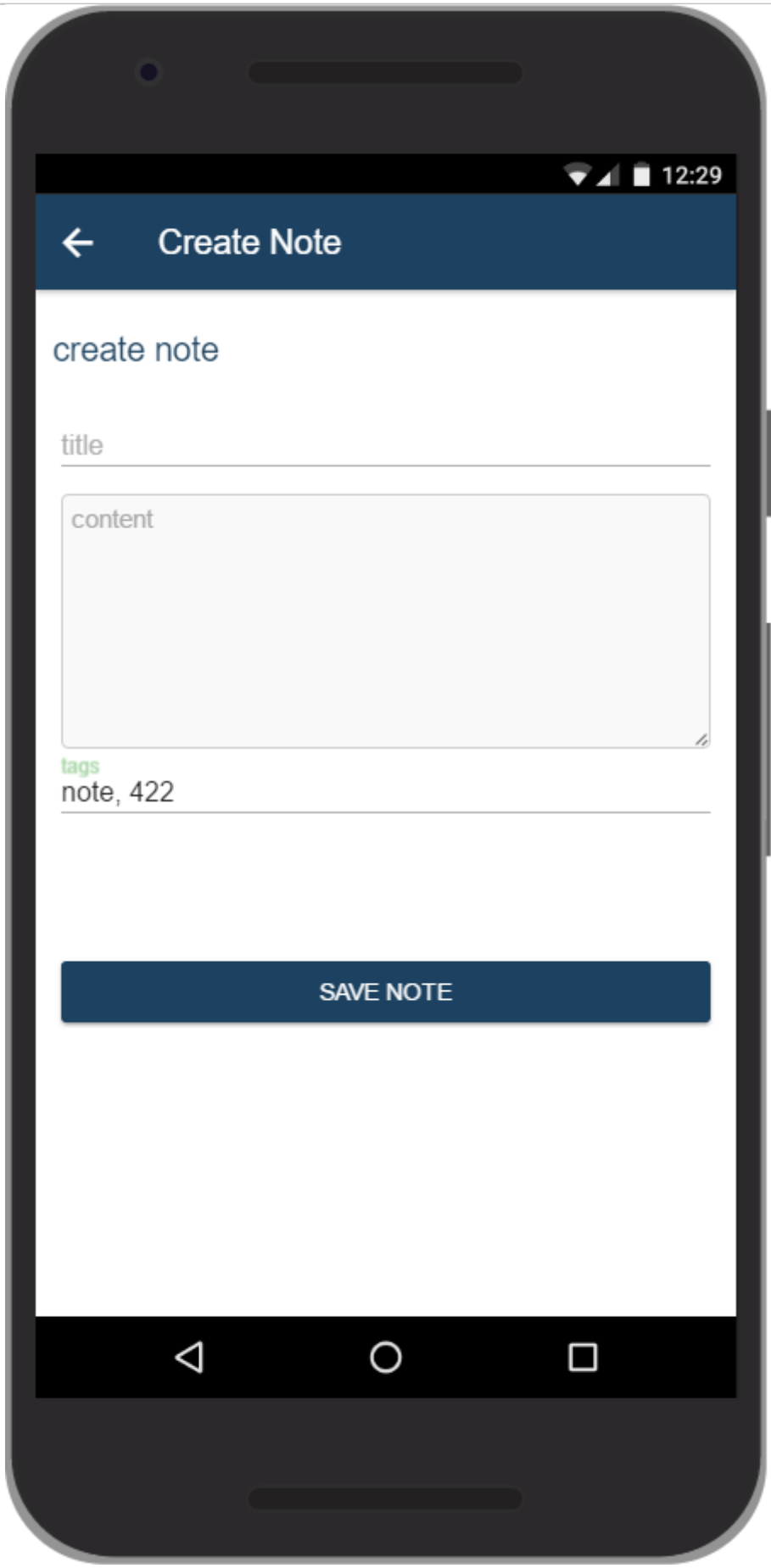
notes app - create note page

- 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

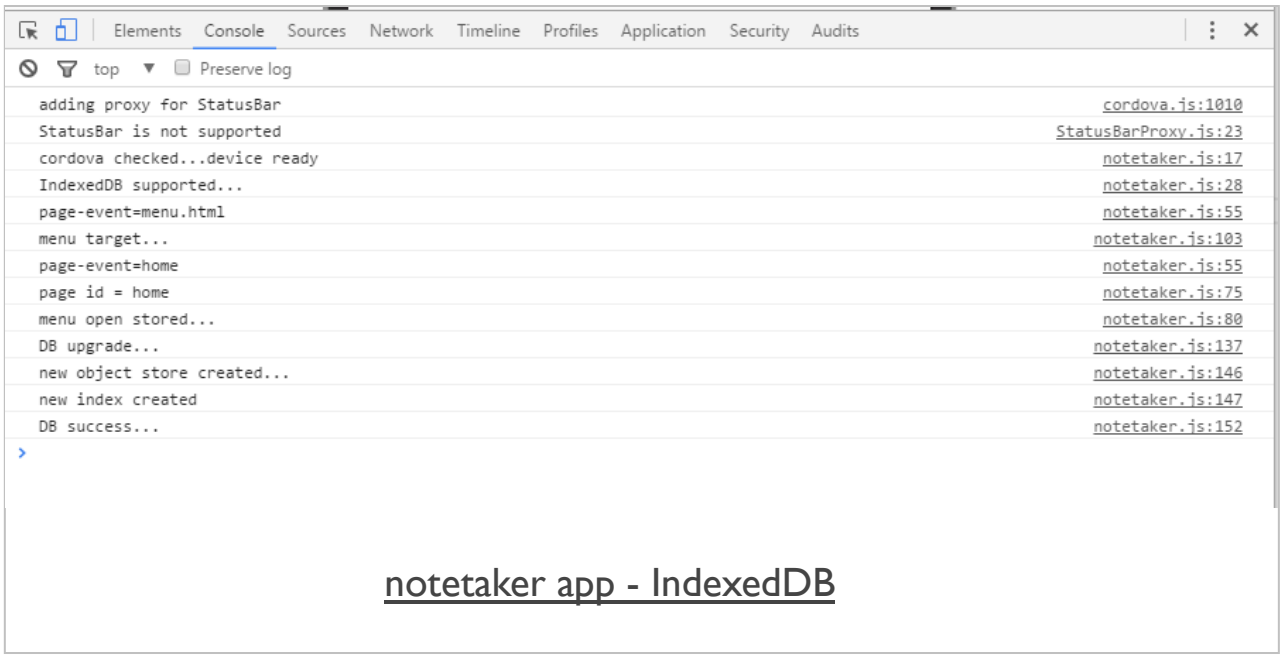
notes app - load initial notes

- 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");
    }
}
...
```

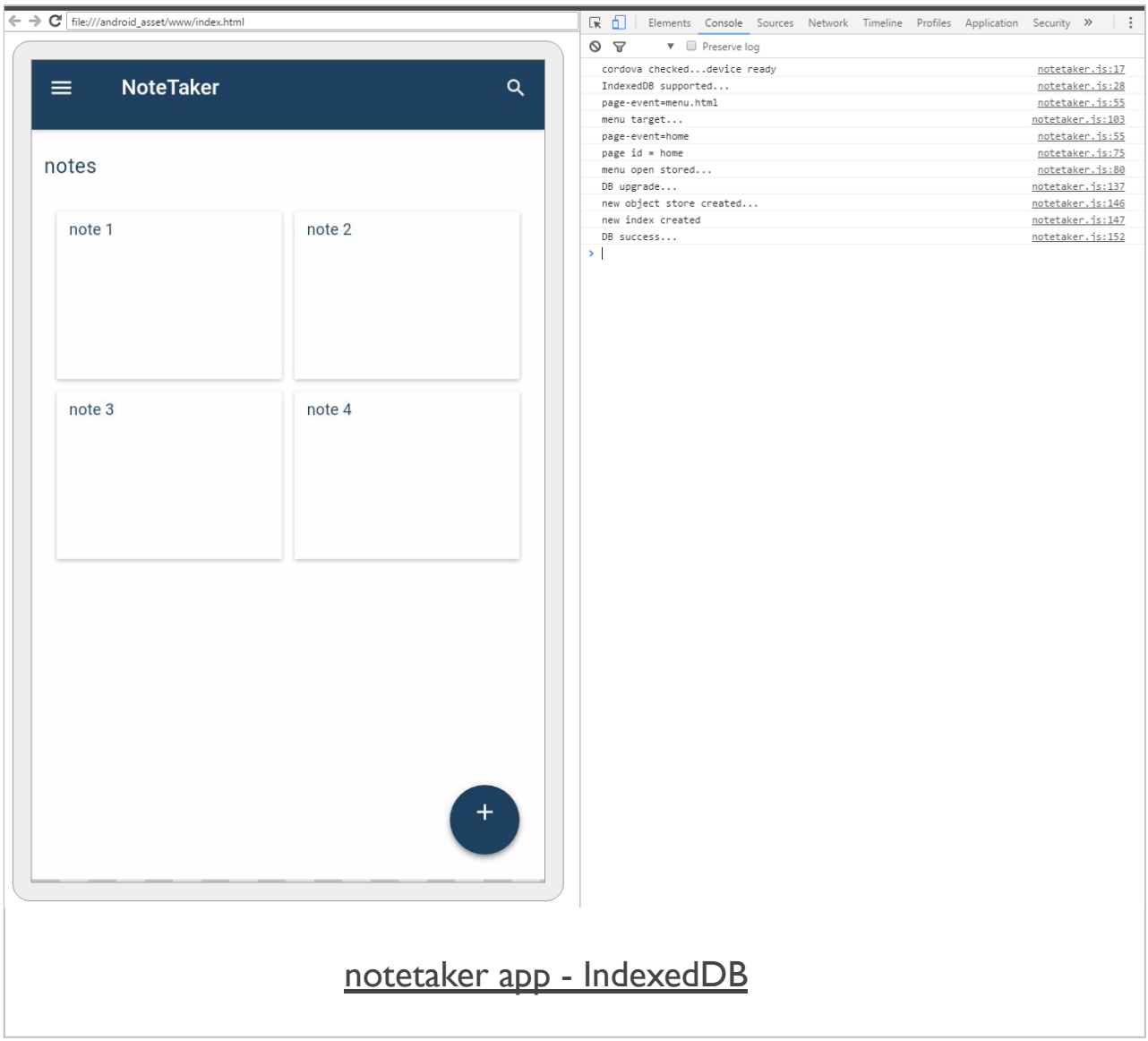
- check a list of existing object stores
- if required object store unavailable we can create our new object store
  - *listen for result from this synchronous method*
- as a user opens our app for the first time
  - *the `upgradeneeded` event is run*
  - *code checks for an existing object store*
 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

Cordova app - NoteTaker - v1 - OnsenUI

notes app - load initial notes

- start to add some data for the initial notes
- IndexedDB allows us to simply store our objects in their default structure
  - *simply store JavaScript objects directly in our IndexedDB database*
- use transactions when working with data and IndexedDB
- transactions help us create a bridge between our app and the current database
  - *allowing us to add our data to the specified object store*
- use the `readwrite` operation on our previous object store, `ntos`

```
var dbTransaction = openDB.transaction(["ntos"],"readwrite");
```

- use it to retrieve the object store for our data

```
var dataStore = dbTransaction.objectStore("ntos");
```

Cordova app - NoteTaker - v1 - OnsenUI

notes app - load initial notes

- set the schema for the note objects

```
// note
var note = {
  title:title,
  note:note,
  tags:tags
}
// add note
var addRequest = dataStore.add(note,key);
```

- schema matches input fields defined for **create note** form

Cordova app - NoteTaker - v1 - OnsenUI

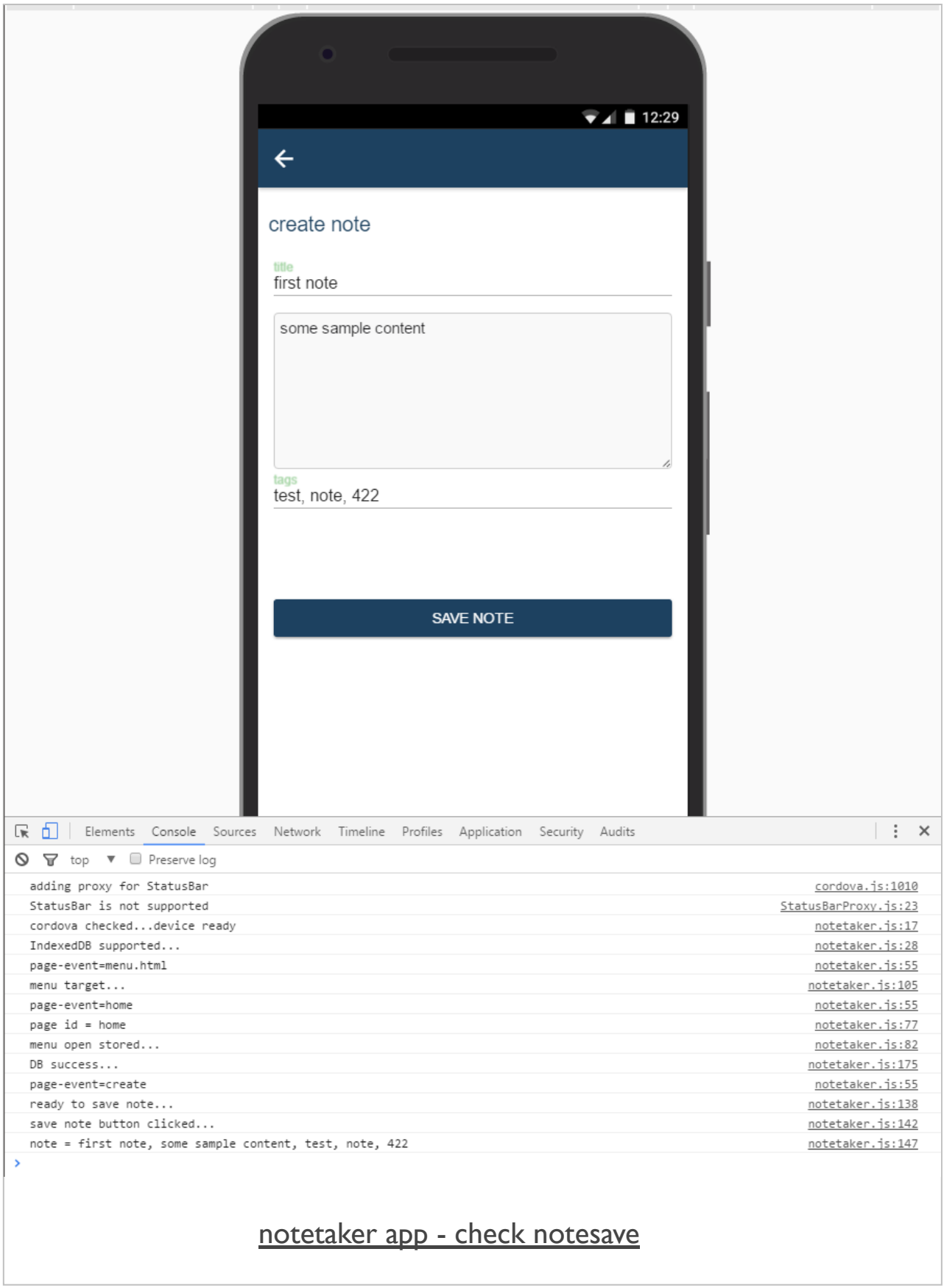
notes app - load initial notes

- add a handler for the **create note** form
- check for the input values
  - *passed to a function to be saved within our database*

```
function createNote(page) {
    //note save handler - check create note page is active...
    if (page.id === "create") {
        console.log("ready to save note...");
        document.getElementById('noteSave').addEventListener('click', function(event) {
            //prevent any bound defaults
            event.preventDefault();
            console.log("save note button clicked...");
            //get values for note - title, content, tags
            var noteTitle = document.getElementById('noteTitle').value;
            var noteContent = document.getElementById('noteContent').value;
            var noteTags = document.getElementById('noteTags').value;
            console.log("note = "+noteTitle+", "+noteContent+", "+noteTags);
      saveNote(noteTitle, noteContent, noteTags);
        });
    }
}
```

- using page events within the app
  - *need to check **create note** page is active, available in the DOM*
  - *before we can start to add listeners for events*
  - *if not, error thrown for the **notesave** element*
- then get values for input fields
- need to validate input before form submission...

Image - check and load IndexedDB



notetaker app - check notesave

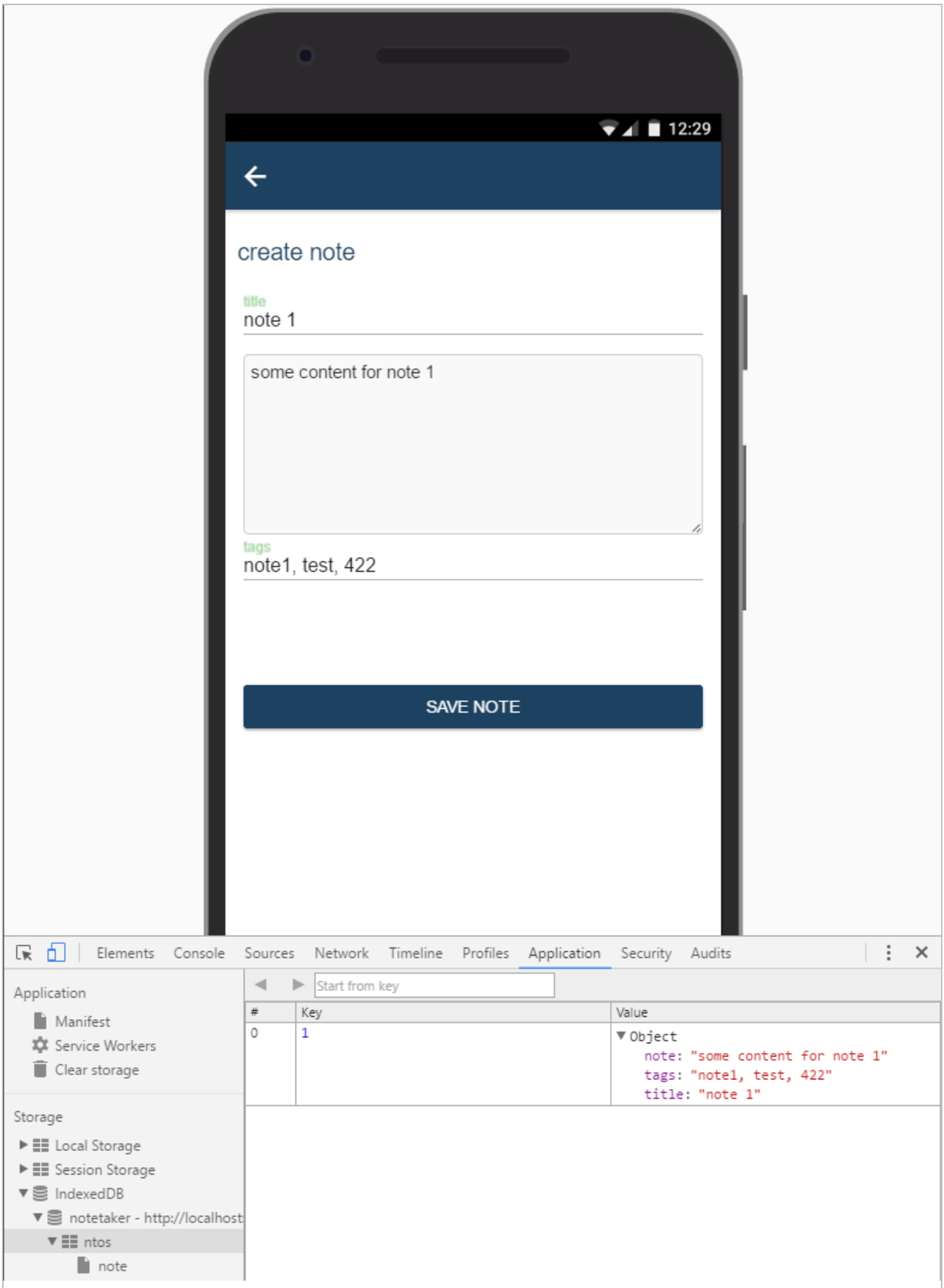
Cordova app - NoteTaker - v1 - OnsenUI

notes app - load initial notes

- add our `saveNote()` function
  - *saves note values in specified object store in the database*

```
//save note data to indexeddb
function saveNote(title, content, tags){
    //define a note
    var note = {
        title:title,
      note:content,
        tags:tags
    }
   // create transaction
    var dbTransaction = db.transaction(["ntos"],"readwrite");
    // define data object store
    var dataStore = dbTransaction.objectStore("ntos");
    // add data to store
    var addRequest = dataStore.add(note);
    // success handler
    addRequest.onsuccess = function(e) {
      console.log("data stored...");
      // do something...
    }
    // error handler
    addRequest.onerror = function(e) {
    console.log(e.target.error.name);
      // handle error...
    }
}
```

Image - check and load IndexedDB



notetaker app - save a note

Cordova app - NoteTaker - v1 - OnsenUI

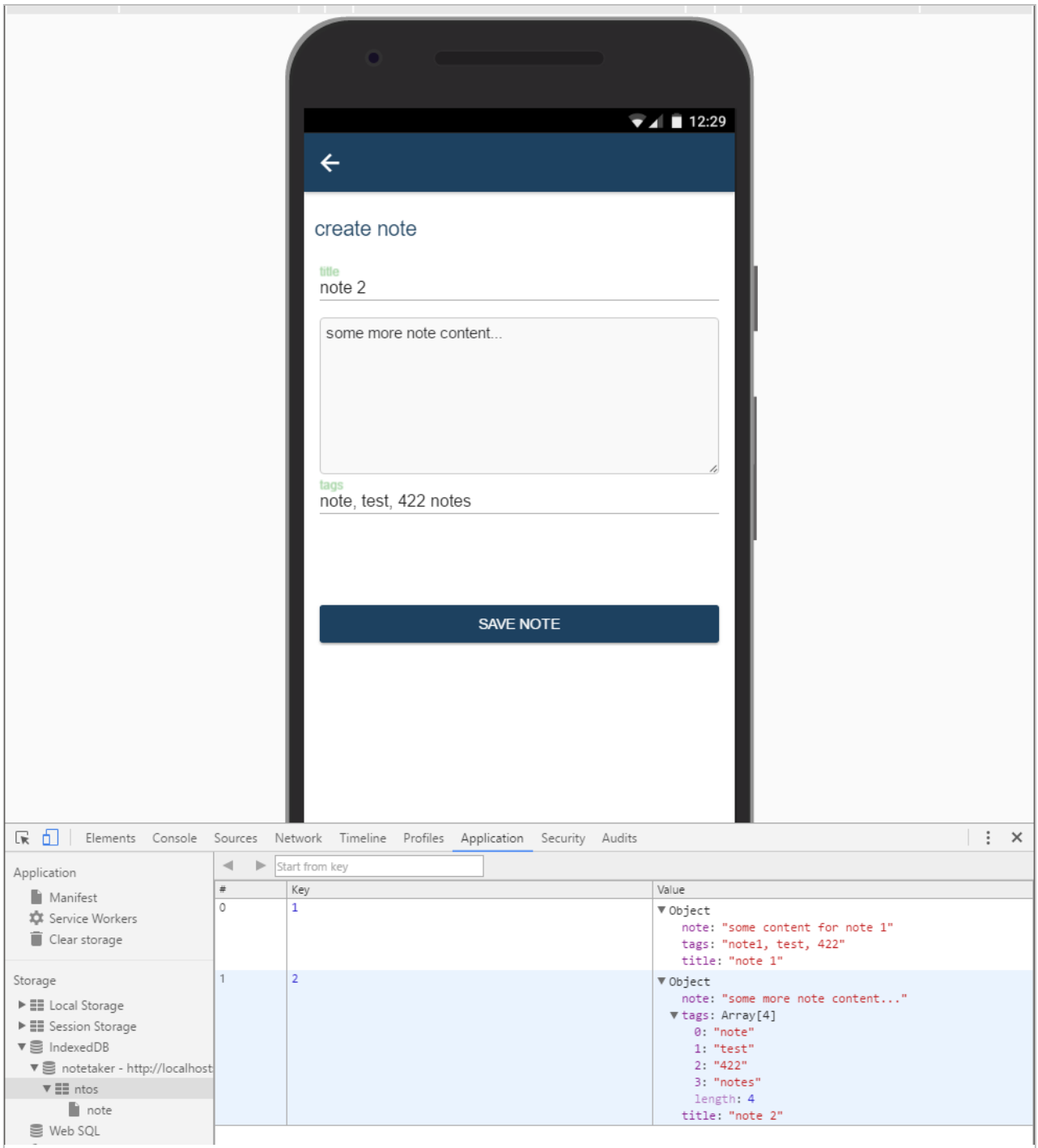
notes app - load initial notes

- need to decide what to do with
  - *tags, success handler, rendering of the new note...*
- as part of the validation for the **tags** input field
  - *we can control structure of input text for tags*
  - *e.g. comma separated, spaces...*
- split each tag from our string
  - *use JS function `split()`*
  - *combine it with a regular expression*

```
...
tags.split(/[ ,]+/);
...
```

- now split our tags from the **create note** form
  - *based on sequence of one or more commas or spaces*

Image - check and load IndexedDB



notetaker app - split tags

Cordova app - NoteTaker - v1 - OnsenUI

notes app - load initial notes

- consider how to handle success event
  - *saving our note data in database's object store*
- might simply return a user to the **create note** form
  - *after showing a notification &c. to provide feedback for saving the note...*
- might return the user to the home page
  - *new note rendered with a feedback message*
- common factors for rendering include the following,
  - *feedback to a user to inform them*
  - *e.g. whether the note was successfully saved or not*
  - *consistent rendering of the notification, buttons, location...*
- consider how to handle error event

Cordova app - NoteTaker - v1 - OnsenUI

notes app - load initial notes

- choose a pattern for **success** event - a saved a note in the database
- show notification with two options
  - ***return to notes*** *and* ***create new note***
- many different patterns available relative to app requirements
- option one - **return to note**
  - *listen for the button's click event*
  - *then dismiss the notification*
  - *pop the* ***create note*** *page from the navigation stack*
  - *return to the home page*
- option two - **create new note**
  - *listen for the button's click event*
  - *dismiss the notification*
  - *reset the form fields*

Cordova app - NoteTaker - v1 - OnsenUI

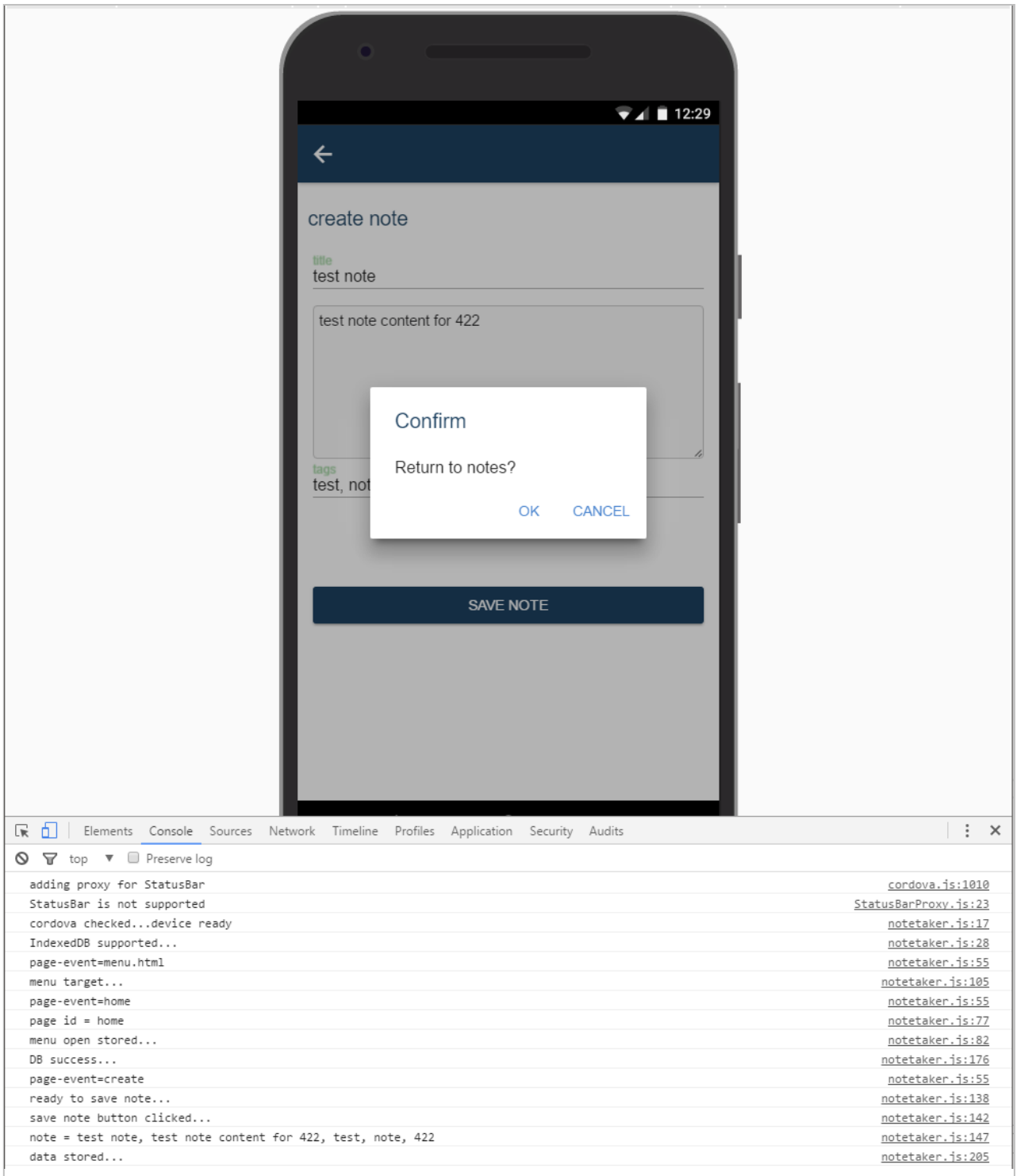
notes app - load initial notes

- add the first option, **return to notes**
  - *as response to user successfully saving a new note*
- need to update the `saveNote()` function

```
...
// success
addRequest.onsuccess = function(e) {
  console.log("data stored...");
  //update user on note stored
  ons.notification.confirm('Return to notes?') // check with user
  .then(function(index) {
    if (index === 1) { // 'ok' button
        document.querySelector('#navigator').popPage(); // return to previous page
    }
  });
}
...
```

- add a notification to Onsen's `ons` object
  - *define it as a confirm notification with message text*
- check user response from button index
- options include
  - *continue with additional new notes*
  - *return to all notes - home page*

Image - save note success



notetaker app - save note success confirmation

Cordova app - NoteTaker - v1 - OnsenUI

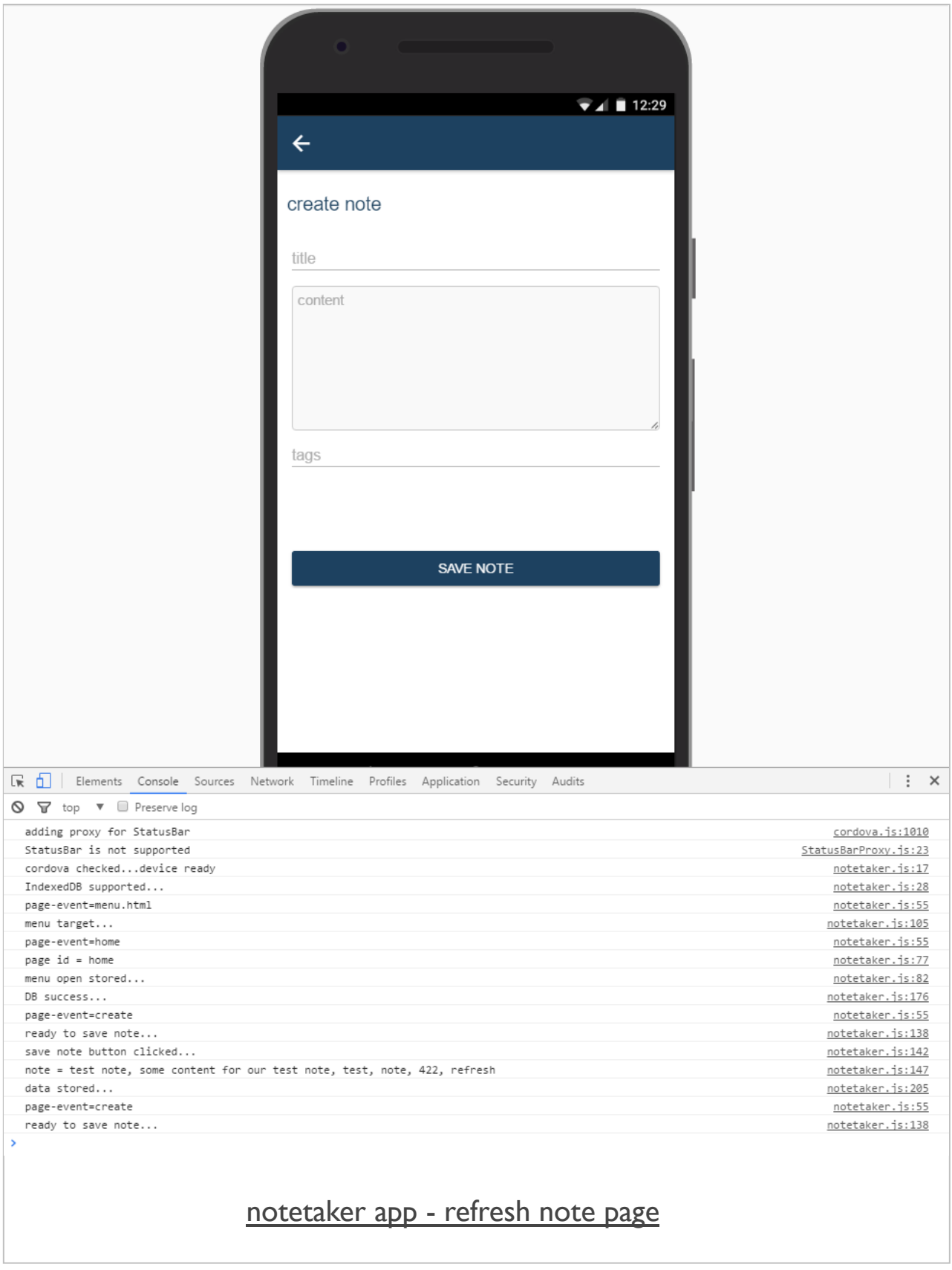
notes app - load initial notes

- need to consider how to handle the current **create note** page
  - *assuming a user selects **cancel** option in the confirmation window*
- user will be returned to the current page in the navigator stack
  - *our **create note** page*
  - *input fields still show previous entry data for note*
- need to ensure that these input fields are cleared
- use existing OnsenUI navigator object to refresh current page

```
//update user on note stored
ons.notification.confirm('Return to notes?') // check with user
.then(function(index) {
    if (index === 1) { // 'ok' button
        document.querySelector('#navigator').popPage(); // return to previous page
  } else if (index === 0) { // check 'cancel' button
        document.querySelector('#navigator').replacePage('create.html', {'animation': 'none'});
    }
});
```

- replace the current top page
  - *set animation for this event to none*

Image - save note success

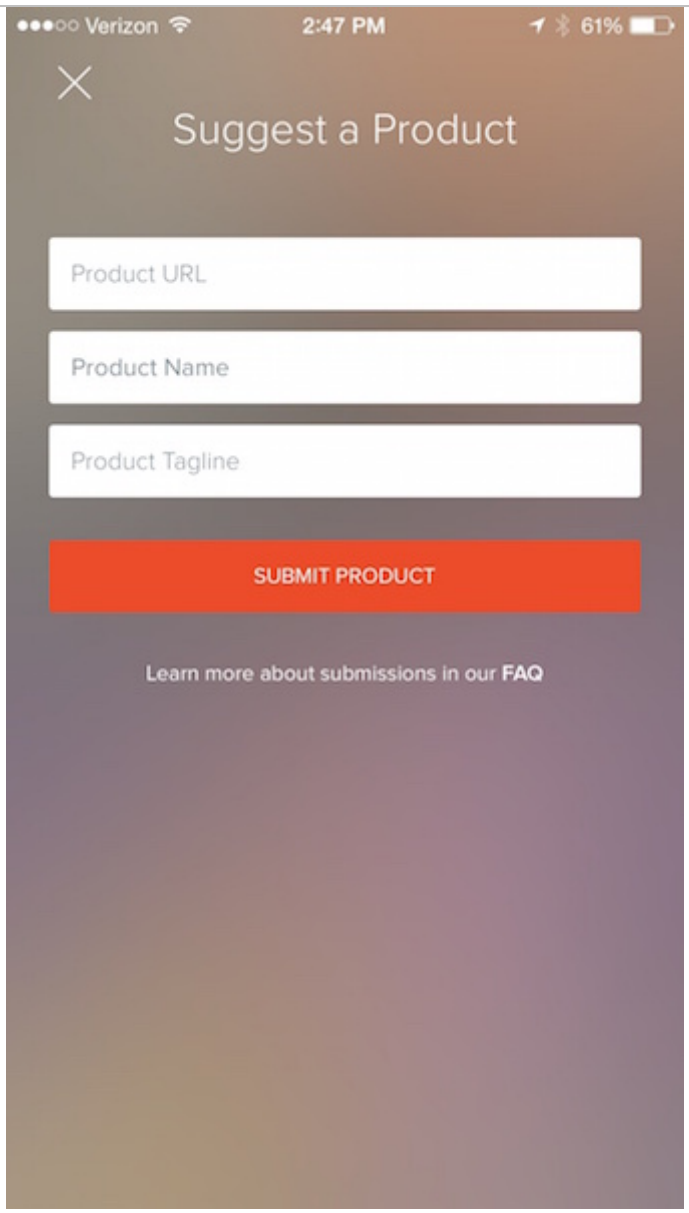


notetaker app - refresh note page

Considering mobile design patterns - forms and data

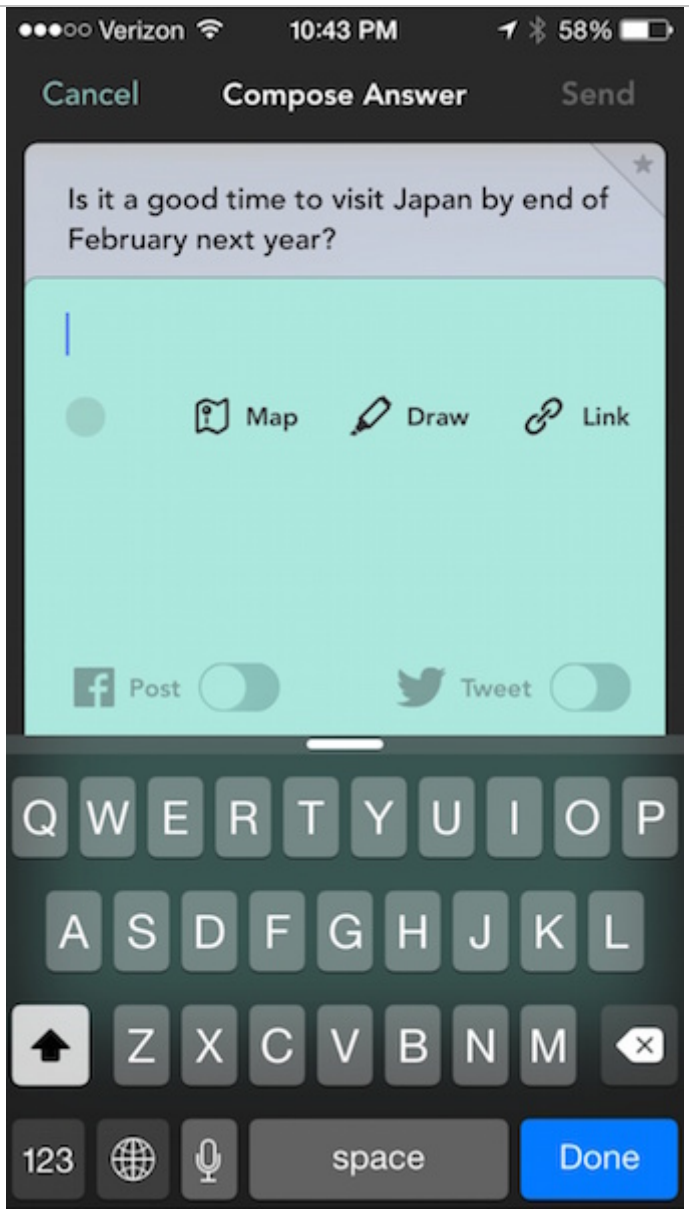
Consider various UI representations of data, including mobile options for forms and data input.

screen 1 - Product Hunt on iOS



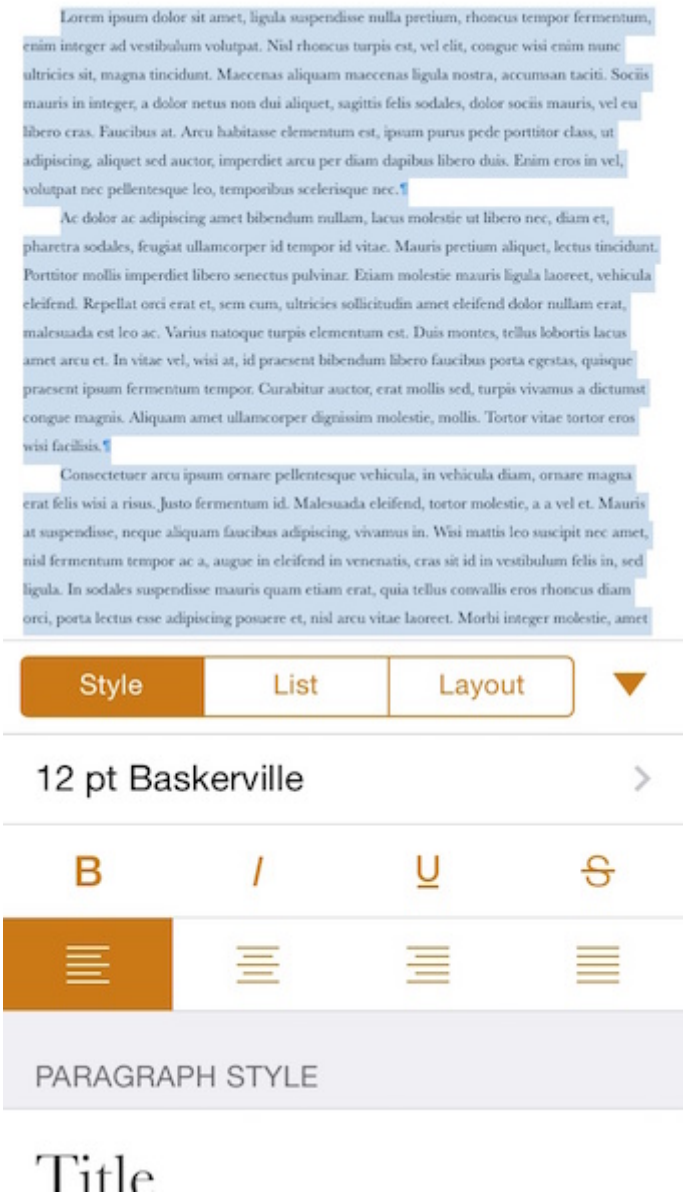
Product Hunt on iOS

screen 2 - Jelly on iOS



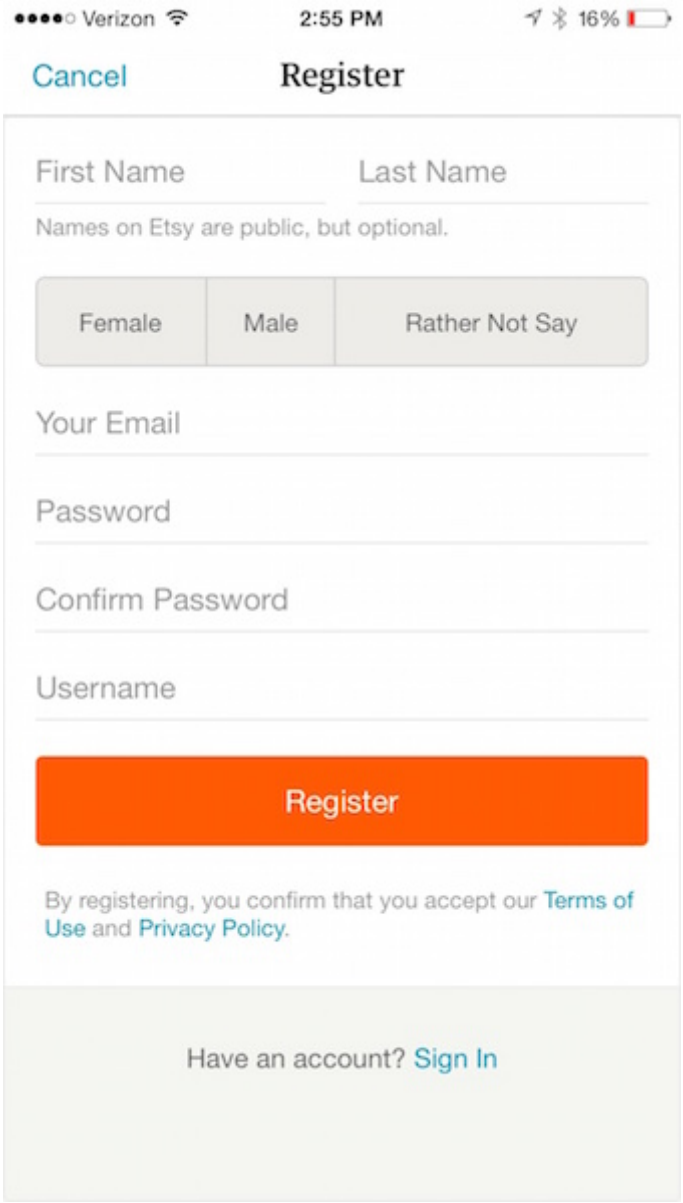
Jelly on iOS

screen 3 - Pages on iOS



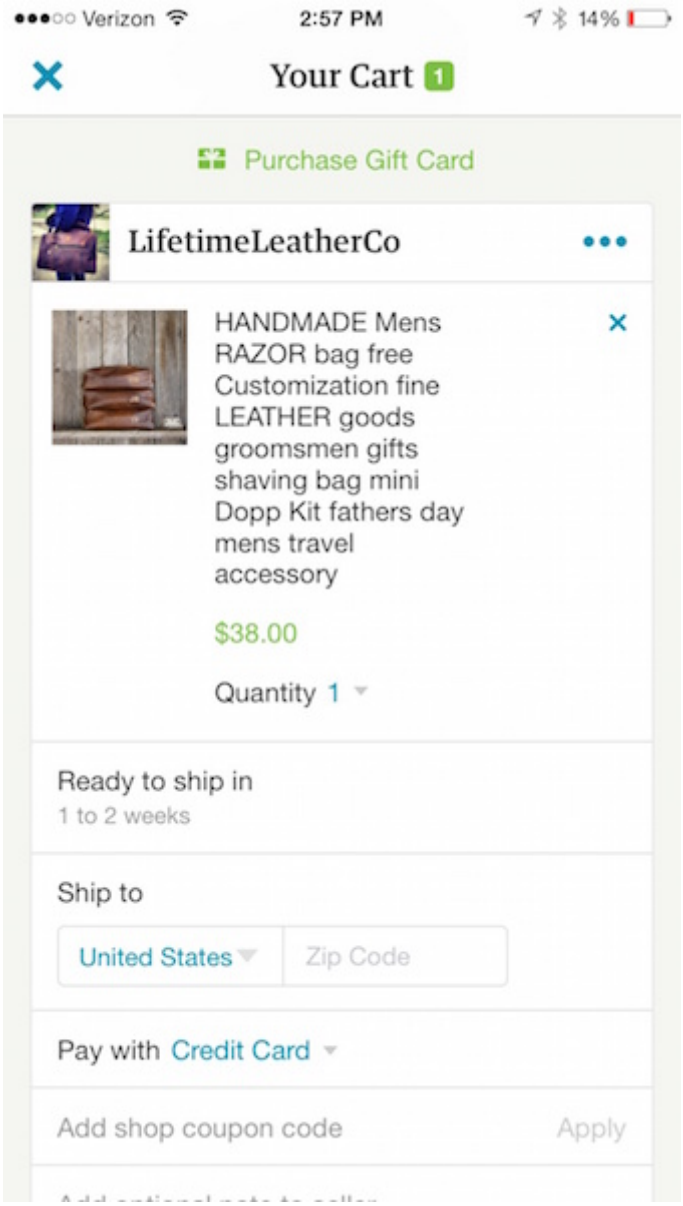
Pages on iOS

screen 4 - eBay on iOS



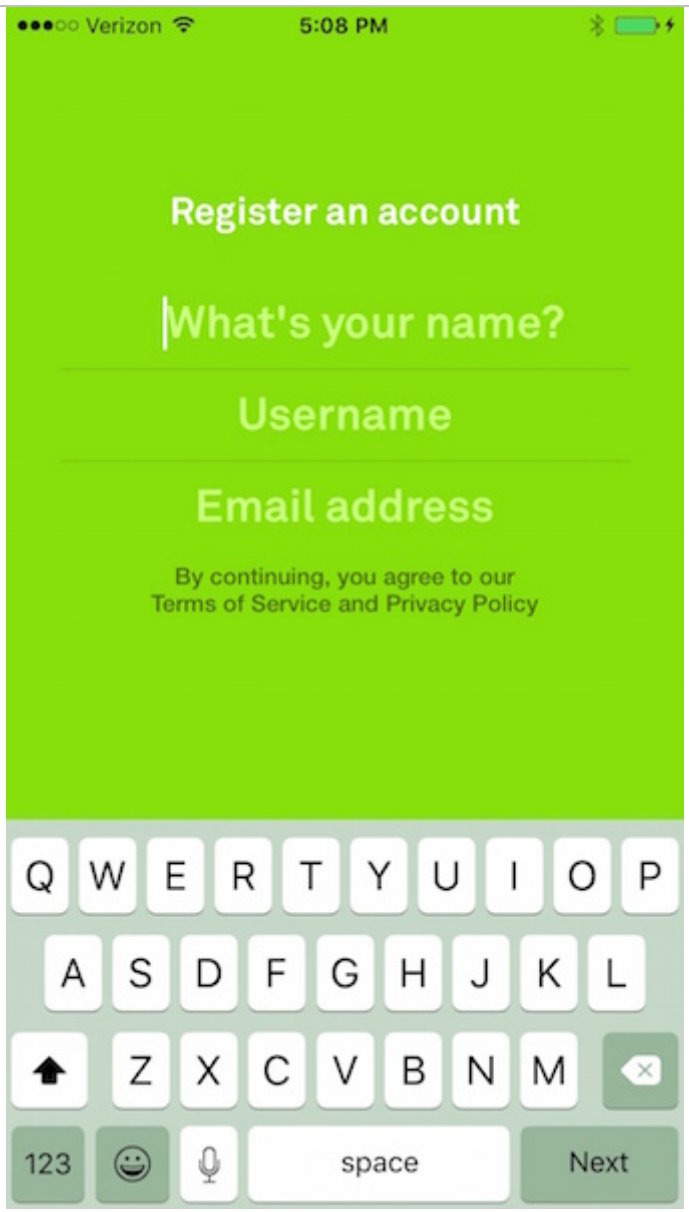
eBay signup on iOS

screen 5 - Etsy on iOS



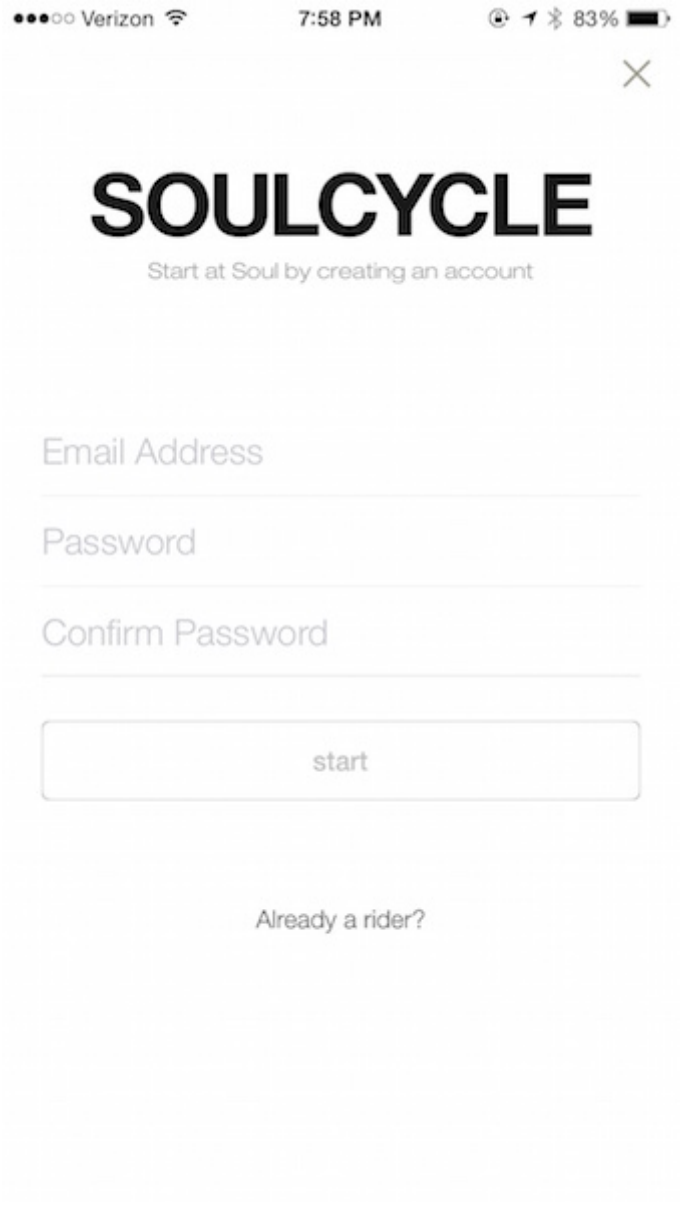
Etsy on iOS

screen 6 - Free on iOS



Free on iOS

screen 7 - SoulCycle on iOS



SoulCycle on iOS

References

- OnsenUI
  - *JavaScript Reference*
- MDN - IndexedDB
  - *IndexedDB API*
- MDN - JavaScript reference
  - *String.prototype.split()*
  - *RegExp*