

Yulp!: Navigation and Routing

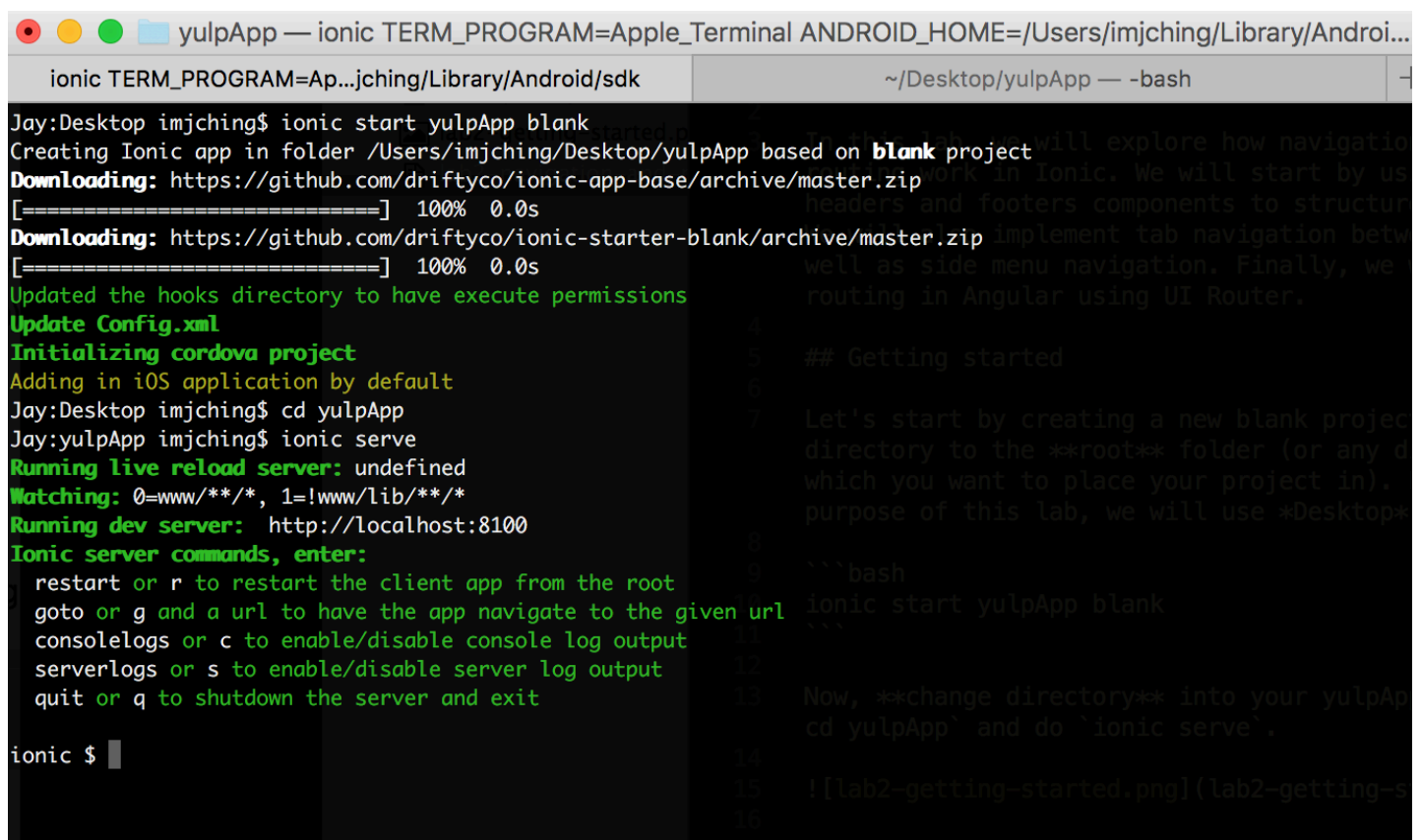
In this lab, we will explore how navigation and routing work in Ionic. We will start by using Ionic's headers and footers components to structure the views. We will also implement tab navigation between views as well as side menu navigation. Finally, we will explore routing in Angular using UI Router.

Getting started

Let's start by creating a new blank project. Change directory to the **root** folder (or any directory in which you want to place your project in). For the purpose of this lab, we will use *Desktop*.

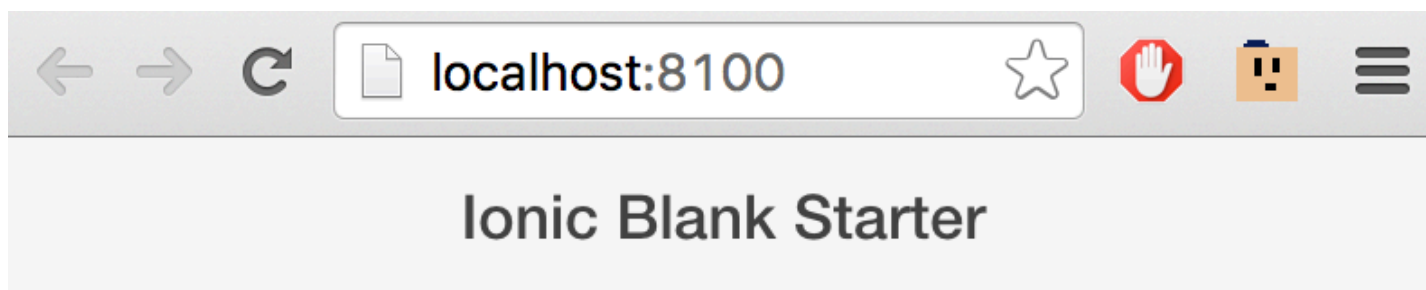
```
ionic start yulpApp blank
```

Now, **change directory** into your yulpApp by doing `cd yulpApp` and do `ionic serve`.



```
ionic TERM_PROGRAM=Apple_Terminal ANDROID_HOME=/Users/imjching/Library/Android...
ionic TERM_PROGRAM=Ap...jching/Library/Android/sdk ~/Desktop/yulpApp — -bash
Jay:Desktop imjching$ ionic start yulpApp blank
Creating Ionic app in folder /Users/imjching/Desktop/yulpApp based on blank project
Downloading: https://github.com/driftyco/ionic-app-base/archive/master.zip
[=====] 100% 0.0s
Downloading: https://github.com/driftyco/ionic-starter-blank/archive/master.zip
[=====] 100% 0.0s
Updated the hooks directory to have execute permissions
Update Config.xml
Initializing cordova project
Adding in iOS application by default
Jay:Desktop imjching$ cd yulpApp
Jay:yulpApp imjching$ ionic serve
Running live reload server: undefined
Watching: 0=www/**/*, 1=!www/lib/**/*
Running dev server: http://localhost:8100
Ionic server commands, enter:
  restart or r to restart the client app from the root
  goto or g and a url to have the app navigate to the given url
  consolelogs or c to enable/disable console log output
  serverlogs or s to enable/disable server log output
  quit or q to shutdown the server and exit
ionic $
```

You will see the following in your browser:



Setting up the tabs

Head to **www/index.html** and change `ng-app="starter"` to `ng-app="yulpApp"`. Then replace the contents of the body tag so that they look like these:

```
<body ng-app="yulpApp">
  <!--
    The nav bar that will be updated as we navigate between views.
  -->
  <ion-nav-bar class="bar-stable">
    <ion-nav-back-button>
  </ion-nav-back-button>
  </ion-nav-bar>
  <!--
    The views will be rendered in the <ion-nav-view> directive below
    Templates are in the /templates folder (but you could also
    have templates inline in this html file if you'd like).
  -->
  <ion-nav-view></ion-nav-view>
</body>
```

Next, head to **js/app.js** and replace it with the following:

```
// Ionic Starter App

// angular.module is a global place for creating, registering and retrieving Angular modules
// 'yulpApp' is the name of this angular module example (also set in a <body> attribute in index.html)
// the 2nd parameter is an array of 'requires'
angular.module('yulpApp', ['ionic'])
```

```

.run(function($ionicPlatform) {
  $ionicPlatform.ready(function() {
    if(window.cordova && window.cordova.plugins.Keyboard) {
      // Hide the accessory bar by default (remove this to show the accessory bar above
      the keyboard
      // for form inputs)
      cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);

      // Don't remove this line unless you know what you are doing. It stops the viewport
      // from snapping when text inputs are focused. Ionic handles this internally for
      // a much nicer keyboard experience.
      cordova.plugins.Keyboard.disableScroll(true);
    }
    if(window.StatusBar) {
      StatusBar.styleDefault();
    }
  });
})

.config(function($stateProvider, $urlRouterProvider) {

  $stateProvider
    .state('home', {
      url: '/home',
      templateUrl: 'views/home/home.html'
    });

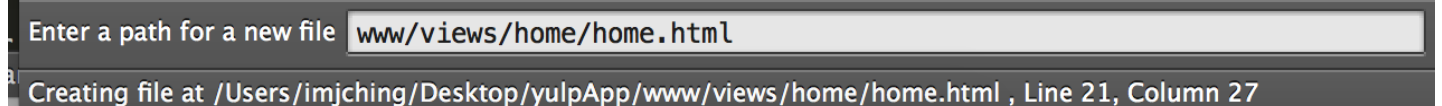
  // if none of the above states are matched, use this as the fallback
  $urlRouterProvider.otherwise('/home');
});

```

Do not worry about the *run* function. You can take it as some boilerplate code to ensure that Cordova runs properly.

Next, create the following file: `views/home/home.html`. The `home.html` should be placed in the home folder of views folder of the www folder.

TIP: If you are using Sublime Text 2 or 3, you can install a plugin called AdvancedNewFile (<https://github.com/skuroda/Sublime-AdvancedNewFile>). This plugin allows you to directly create a file even though the folder do not exists.



In `home.html`, paste the following code into it:

```
<!--
```

Create tabs with an icon and label, using the *tabs-positive* style.
Each tab's child `<ion-nav-view>` directive will have its own
navigation history that also transitions its views in and out.

icon-off and *icon-on* both represent the icons that are used when the tabs
are being checked or unchecked.

```
-->
```

```
<ion-tabs class="tabs-icon-only tabs-color-active-positive tabs-icon-only">
```

```
  <!-- Feed Tab -->
```

```
  <ion-tab title="Feed" icon-off="ion-ios-home-outline" icon-on="ion-ios-home" href="#">
```

```
    <ion-nav-view></ion-nav-view>
```

```
  </ion-tab>
```

```
  <!-- Search Tab -->
```

```
  <ion-tab title="Search" icon-off="ion-ios-search" icon-on="ion-ios-search-strong" href="#">
```

```
    <ion-nav-view></ion-nav-view>
```

```
  </ion-tab>
```

```
  <!-- Settings Tab -->
```

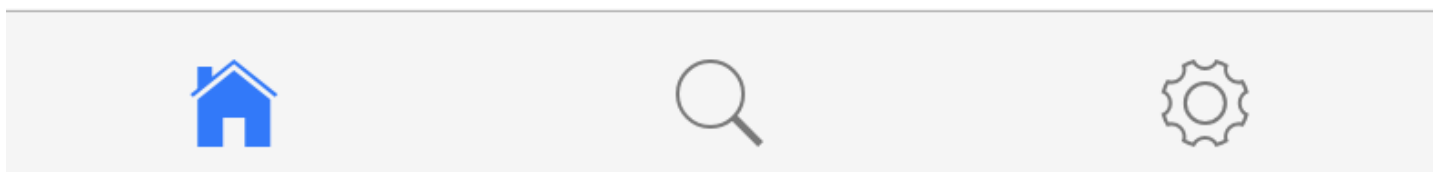
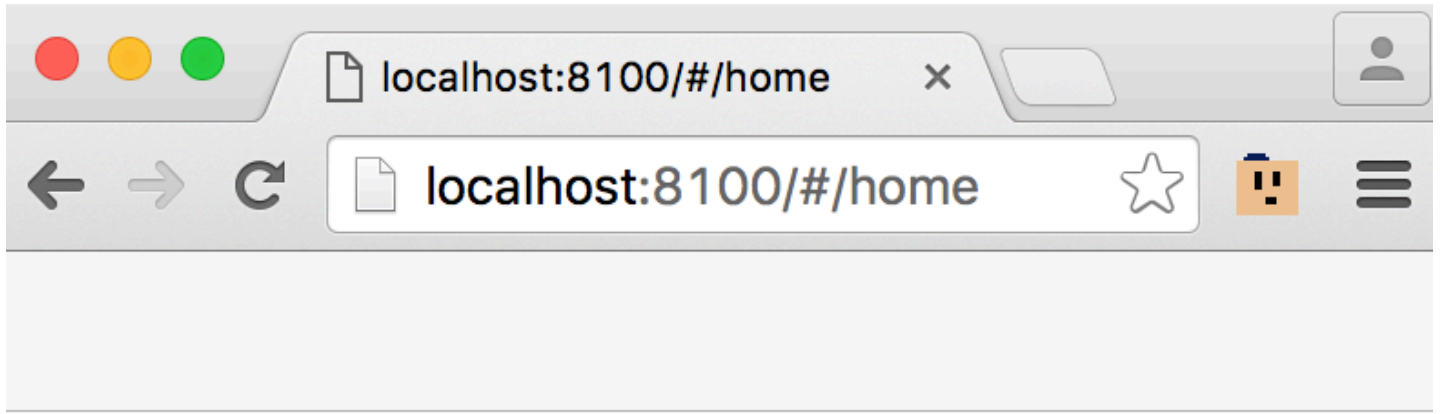
```
  <ion-tab title="Settings" icon-off="ion-ios-gear-outline" icon-on="ion-ios-gear" href="#">
```

```
    <ion-nav-view></ion-nav-view>
```

```
  </ion-tab>
```

```
</ion-tabs>
```

Now, your app should look like this:



Adding routing for tabs

Now let's go back to `app.js` and add the respective routes for the tabs.

We will first begin by modifying the home state and making it abstract.

```
$stateProvider
  .state('home', {
    url: '/home',
    templateUrl: 'views/home/home.html',
    abstract: true
  });
```

When you go back to the application, you will see a blank screen. As stated before, if a state is abstract, we would not be able to navigate to it.

Head to `home.html` and add the `name` property to the `ion-nav-view` tags.

```
<ion-tabs class="tabs-icon-only tabs-color-active-positive tabs-icon-only">

  <!-- Feed Tab -->
  <ion-tab title="Feed" icon-off="ion-ios-home-outline" icon-on="ion-ios-home" href="#">
    <!-- tab-feed is the name of the template that you are going to render -->
    <ion-nav-view name="tab-feed"></ion-nav-view>
  </ion-tab>

  <!-- Search Tab -->
  <ion-tab title="Search" icon-off="ion-ios-search" icon-on="ion-ios-search-strong" href="#">
    <!-- tab-search is the name of the template that you are going to render -->
    <ion-nav-view name="tab-search"></ion-nav-view>
  </ion-tab>

  <!-- Settings Tab -->
  <ion-tab title="Settings" icon-off="ion-ios-gear-outline" icon-on="ion-ios-gear" href="#">
    <!-- tab-settings is the name of the template that you are going to render -->
    <ion-nav-view name="tab-settings"></ion-nav-view>
  </ion-tab>

</ion-tabs>
```

We have `tab-feed`, `tab-search`, and `tab-settings`. Let's add routes for them. Modify `app.js` so that it looks like this:

```
// some code before this...
```

```

.config(function($stateProvider, $urlRouterProvider) {

  $stateProvider
    .state('home', {
      url: '/home',
      templateUrl: 'views/home/home.html',
      abstract: true
    })
    .state('home.feed', {
      url: '/feed', // url will be /home/feed
      views: {
        'tab-feed': {
          templateUrl: 'views/home/feed.html'
        }
      }
    })
    .state('home.search', {
      url: '/search', // url will be /home/search
      views: {
        'tab-search': {
          templateUrl: 'views/home/search.html'
        }
      }
    })
    .state('home.settings', {
      url: '/settings', // url will be /home/settings
      views: { // named views
        'tab-settings': {
          templateUrl: 'views/home/settings.html'
        }
      }
    });

  // as you can see, home.feed, home.search and home.settings
  // have a parent home

  // if none of the above states are matched, use this as the fallback
  $urlRouterProvider.otherwise('/home/feed');
});

```

Next, create the following views:

- views/home/feed.html

```

<ion-view view-title="Yulp!">
  <ion-content>
    <h1>Todo: News Feed here...</h1>
  </ion-content>
</ion-view>

```

- views/home/search.html

```
<ion-view view-title="Search">
  <ion-content>
    <h1>Todo: Search here...</h1>
  </ion-content>
</ion-view>
```

- views/home/settings.html

```
<ion-view view-title="Settings">
  <ion-content>
    <ion-list>

      <ion-toggle>
        Sample Toggle
      </ion-toggle>

    </ion-list>
  </ion-content>
</ion-view>
```

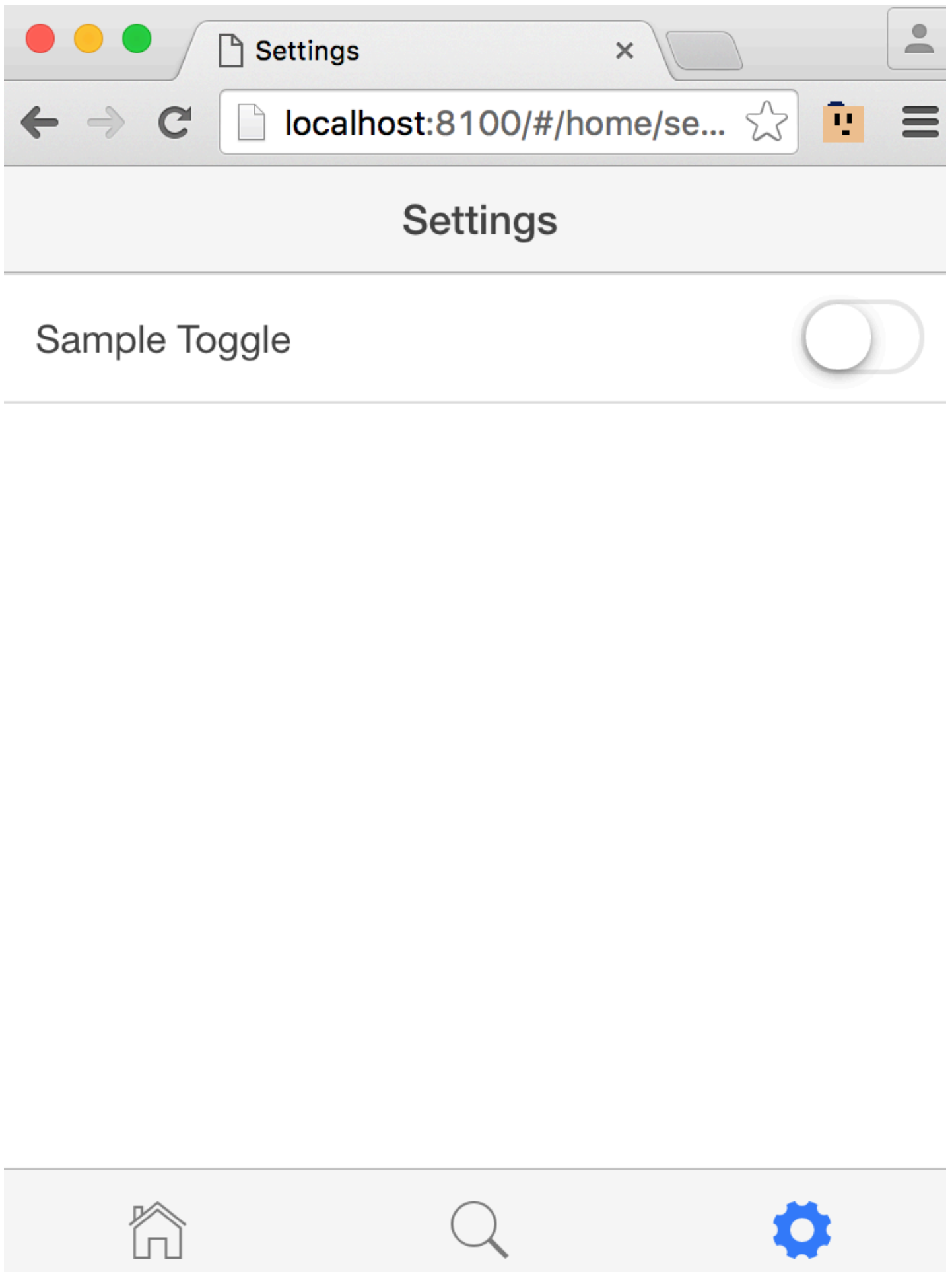
We will also need to modify the `href` property of our tab links in `home.html` .

```
...
<ion-tab title="Feed" .... href="#/home/feed">
...
<ion-tab title="Search" ... ui-sref="home.search">
...
<ion-tab title="Settings" ... ui-sref="home.settings">
...
```

Noticed that we use both methods `href` and `ui-sref` . You can pick any one of them. However, it is suggested to use `ui-sref` as it would be easier to handle in the long run.

dot-dot-dot represents code snippets that have been omitted.

You should be able to navigate between views when you click on the respective tabs.



We are done for now. We will start designing the respective tab layouts when we deal with Ionic Components in the next lab.

Extra time?

Head to <http://ionicframework.com/docs/components/>.

Take a look at the following components and try to play around with them (by pasting it into `settings.html`). Don't worry about messing up with the code, we will replace the whole contents of *settings.html* later on.

Reading

Notice this snippet in `settings.html` ?

```
<ion-toggle>
  Sample Toggle
</ion-toggle>
```

Ref: <http://ionicframework.com/docs/api/directive/ionToggle/>

This is essentially the **same** as this:

```
<li class="item item-toggle">
  Sample Toggle
  <label class="toggle">
    <input type="checkbox">
    <div class="track">
      <div class="handle"></div>
    </div>
  </label>
</li>
```

Ref: <http://ionicframework.com/docs/components/#toggle>

ion-toggle is an Angular custom directive made specifically for Ionic Framework. It is easier to use and it enhances the readability of your code.