CS 385 Lecture 10 – interactive with Lab 4

There is NO Moodle Quiz today

- We need to take our time to work through the lecture notes today and understand the steps involved.
- This MAY require you to work on this lecture/lab outside of the laboratory hours (on your own time).
- I will provide an OPTIONAL (SELF ASSESSMENT) assignment – for students interested in working with Ionic and API.
- I'll provide a selection of APIs and you are required to develop a NEW Ionic Application to display the API and provide a 'clickable' page.

Getting Started:

- As before -open the Node JS Command Prompt
- Go to your X Drive on the command prompt.
- Create a directory/folder for today's lecture/lab
- Then get ready to begin the process of creating an Ionic project
- **We'll always need our two IONIC Node JS Command Prompt windows opened.**
- **REMEMBER to have your command prompts INSIDE your project folder.**

Getting Started. Create a new project

Node.js command prompt

```
X:\IONIC\lab4a>ionic start ionicLab4 blank
```

Command: **ionic start ionicLab4 blank**

npm

```
X:\IONIC\lab4a>ionic start ionicLab4 blank
√ Preparing directory .\ionicLab4 - done!
√ Downloading and extracting blank starter - done!
? Integrate your new app with Cordova to target native iOS and Android? (y/N)
```

```
--> Install DevApp: https://bit.ly/ionic-dev-app <--

> npm i
npm WARN deprecated browserslist@2.11.3: Browserslist 2 could fail on reading Browsers
ls.
npm WARN deprecated hoek@2.16.3: The major version is no longer supported. Please upda
[        ..........] / extract:execa: sill pacote execa@https://registry.npmjs.org/exe
```

Then wait patiently for ionic to create the boilerplate for your application

ERROR – If you get this error DO NOT PANIC – this is nothing serious.

```
npm ERR!   cause:
npm ERR!    { Error: EPERM: operation not permitted, rename 'X:\IONIC\lab4a\ionicL
n.643420238' -> 'X:\IONIC\lab4a\ionicLab4\node_modules\glob-base\package.json'
npm ERR!      errno: -4048,
npm ERR!      code: 'EPERM',
npm ERR!      syscall: 'rename',
npm ERR!      path: 'X:\\IONIC\\lab4a\\ionicLab4\\node_modules\\glob-base\\package
npm ERR!      dest: 'X:\\IONIC\\lab4a\\ionicLab4\\node_modules\\glob-base\\package
npm ERR!   stack: 'Error: EPERM: operation not permitted, rename \'X:\\IONIC\\lab4
\package.json.643420238\' -> \'X:\\IONIC\\lab4a\\ionicLab4\\node_modules\\glob-bas
npm ERR!   errno: -4048,
npm ERR!   code: 'EPERM',
npm ERR!   syscall: 'rename',
npm ERR!   path: 'X:\\IONIC\\lab4a\\ionicLab4\\node_modules\\glob-base\\package.js
npm ERR!   dest: 'X:\\IONIC\\lab4a\\ionicLab4\\node_modules\\glob-base\\package.js
npm ERR!   parent: 'ionicLab4' }
npm ERR!
npm ERR! Please try running this command again as root/Administrator.

npm ERR! A complete log of this run can be found in:
npm ERR!     C:\Users\testuser1\AppData\Roaming\npm-cache\_logs\2018-10-23T13_25_2
[ERROR] Non-zero exit from subprocess.

X:\IONIC\lab4a>
```

Run the command **npm install**

Which will (hopefully fix this)

ERROR – If you get this error DO NOT PANIC – this is nothing serious.

```
X:\IONIC\lab4a>cd ionicLab4

X:\IONIC\lab4a\ionicLab4>dir
 Volume in drive X is testuser1
 Volume Serial Number is 34B0-CE18

 Directory of X:\IONIC\lab4a\ionicLab4

23/10/2018  14:24    <DIR>          .
23/10/2018  14:28    <DIR>          ..
23/10/2018  14:23             6,238 config.xml
23/10/2018  14:23                95 ionic.config.json
16/10/2018  16:25               197 ionic.starter.json
23/10/2018  14:25    <DIR>          node_modules
23/10/2018  14:23             1,088 package.json
23/10/2018  14:23    <DIR>          resources
23/10/2018  14:23    <DIR>          src
16/10/2018  16:25               519 tsconfig.json
16/10/2018  16:25               178 tslint.json
               6 File(s)          8,315 bytes
               5 Dir(s)   6,254,112,768 bytes free

X:\IONIC\lab4a\ionicLab4>npm install
```

Change into the directory which is the name of your project and then run this command

Run the command npm install

Which will (hopefully fix this)

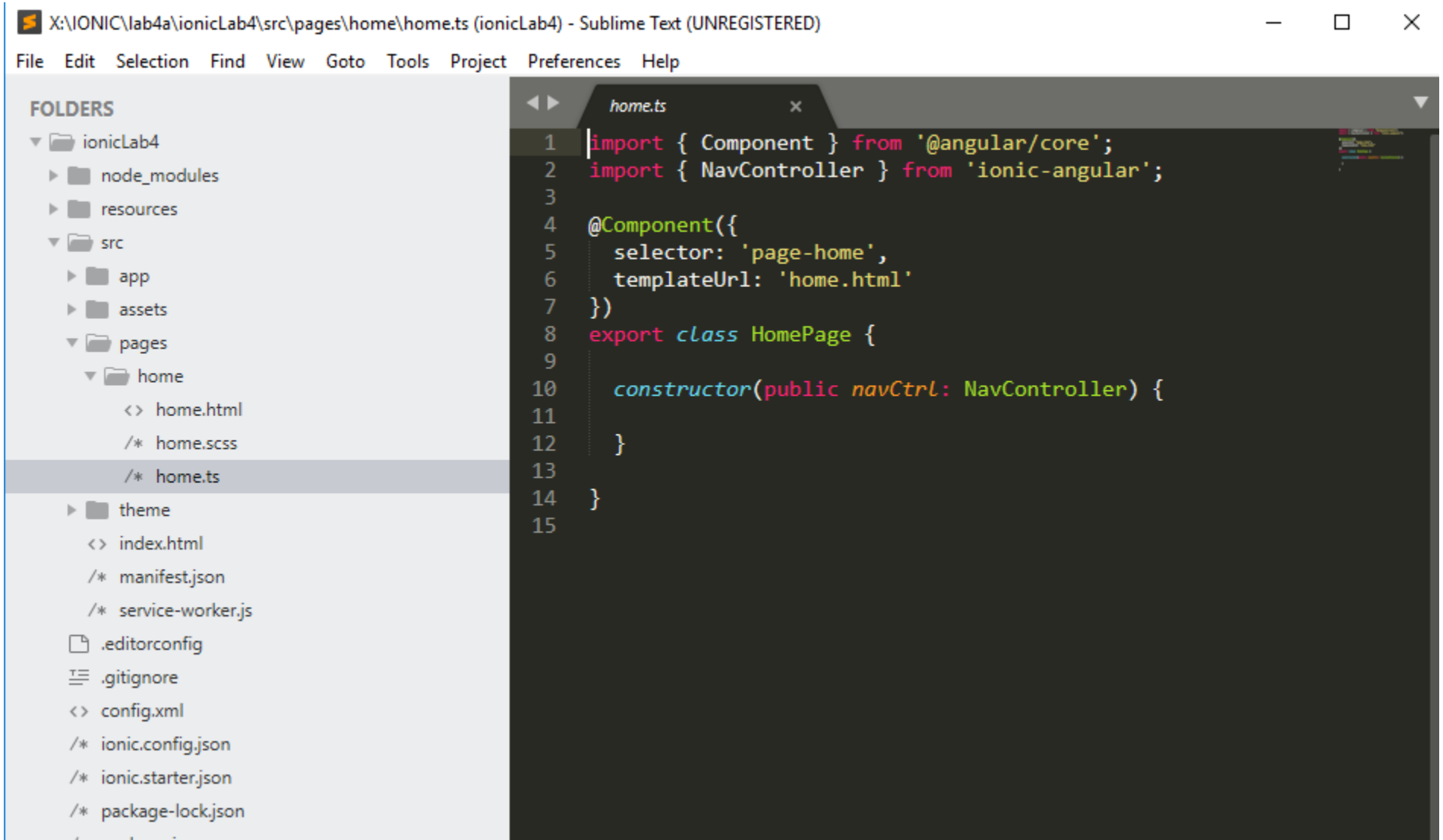
Then wait patiently

And then the Error is resolved and everything is OK.

Node.js command prompt

```
 Directory of X:\IONIC\lab4a\ionicLab4

23/10/2018  14:24    <DIR>          .
23/10/2018  14:28    <DIR>          ..
23/10/2018  14:23             6,238 config.xml
23/10/2018  14:23                95 ionic.config.json
16/10/2018  16:25               197 ionic.starter.json
23/10/2018  14:25    <DIR>          node_modules
23/10/2018  14:23             1,088 package.json
23/10/2018  14:23    <DIR>          resources
23/10/2018  14:23    <DIR>          src
16/10/2018  16:25               519 tsconfig.json
16/10/2018  16:25               178 tslint.json
               6 File(s)          8,315 bytes
               5 Dir(s)   6,254,112,768 bytes free

X:\IONIC\lab4a\ionicLab4>npm install
npm WARN deprecated browserslist@2.11.3: Browserslist 2 could fail on reading Browserslist
ls.
npm WARN rollback Rolling back node-pre-gyp@0.10.0 failed (this is probably harmless): EPER
stat 'X:\IONIC\lab4a\ionicLab4\node_modules\fsevents\node_modules'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.4 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: want
"} (current: {"os":"win32","arch":"x64"})

added 14 packages, removed 1 package and updated 63 packages in 21.706s

X:\IONIC\lab4a\ionicLab4>
```

Open the project folder in Sublime Text 3



Open the second Node.JS command prompt and SERVE the application

Type the command **`ionic serve --lab`**

npm

```
X:\>cd IONIC

X:\IONIC>cd lab4a

X:\IONIC\lab4a>cd ionicLab4

X:\IONIC\lab4a\ionicLab4>ionic serve --lab -c
> ionic-app-scripts serve --address 0.0.0.0 --port 8100 --livereload-port 35729 --dev-logger-port 53703 --consolelogs --
nobrowser
```

```
[app-scripts]     at Object.module.exports.missingBinary (X:\IONIC\lab4a\ionicLab4\node_
45:5)
[app-scripts]     at module.exports (X:\IONIC\lab4a\ionicLab4\node_modules\node-sass\lib
[app-scripts]     at Object.<anonymous> (X:\IONIC\lab4a\ionicLab4\node_modules\node-sass
[app-scripts]     at Module._compile (module.js:652:30)
[app-scripts]     at Object.Module._extensions..js (module.js:663:10)
[app-scripts]     at Module.load (module.js:565:32)

[INFO] Looks like @ionic/app-scripts isn't installed in this project.

       This package is required for this command to work properly.

? Install @ionic/app-scripts? (Y/n)
```

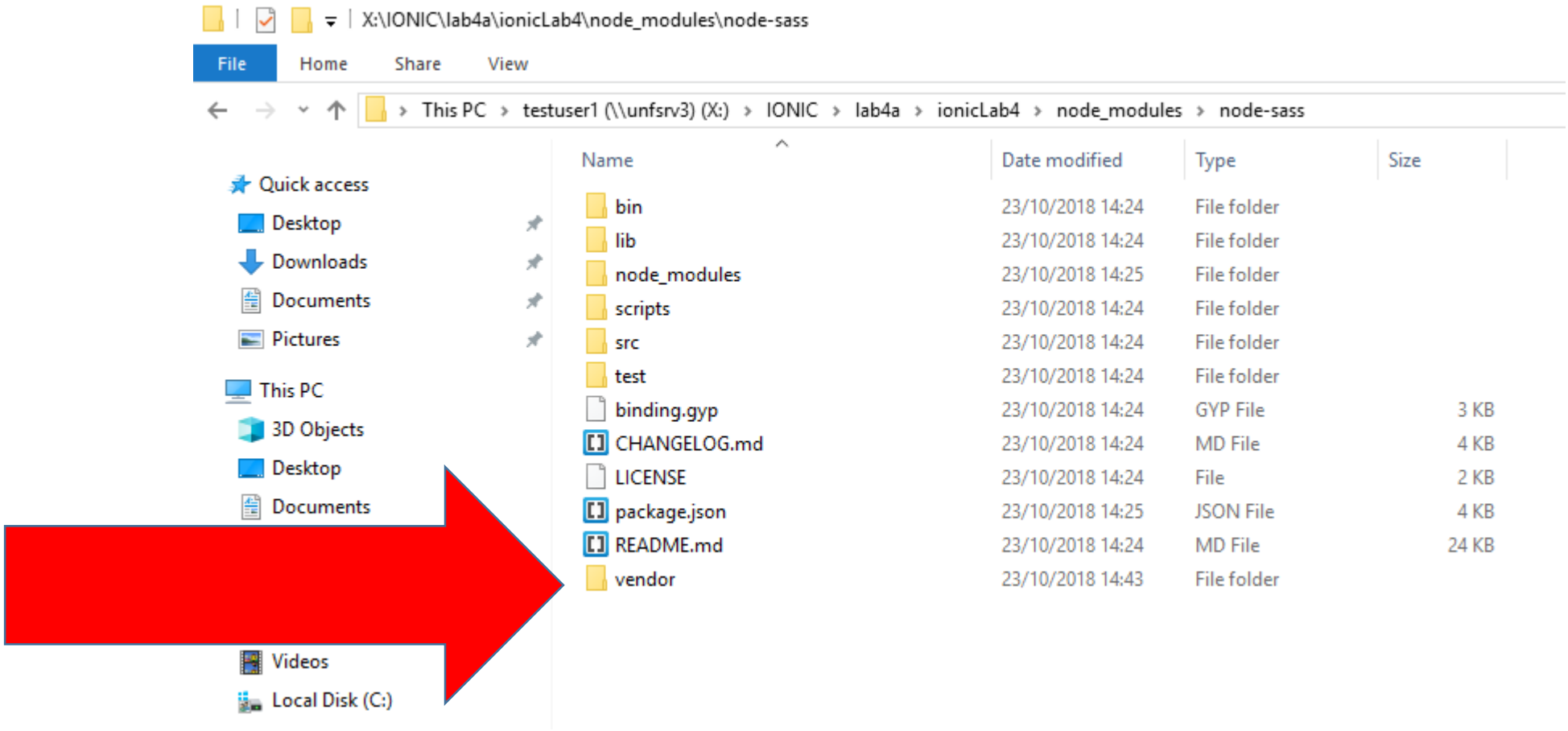
You might be asked to install ionic/app-scripts – answer Y

ERROR 2 (Possible) – regarding node-sass

npm

```
Cached binary found at C:\Users\testuser1\AppData\Roaming\npm-cache\node-sass\4.9.0\win32-x64-57_binding.node

> node-sass@4.9.0 postinstall X:\IONIC\lab4a\ionicLab4\node_modules\node-sass
> node scripts/build.js

Building: C:\Program Files\nodejs\node.exe X:\IONIC\lab4a\ionicLab4\node_modules\node-gyp\bin\node-gyp.js rebuild
ose --libsass_ext= --libsass_cflags= --libsass_ldflags= --libsass_library=
gyp info it worked if it ends with ok
gyp verb cli [ 'C:\\Program Files\\nodejs\\node.exe',
gyp verb cli   'X:\\IONIC\\lab4a\\ionicLab4\\node_modules\\node-gyp\\bin\\node-gyp.js',
gyp verb cli   'rebuild',
gyp verb cli   '--verbose',
gyp verb cli   '--libsass_ext=',
gyp verb cli   '--libsass_cflags=',
gyp verb cli   '--libsass_ldflags=',
gyp verb cli   '--libsass_library=' ]
gyp info using node-gyp@3.8.0
gyp info using node@8.11.4 | win32 | x64
gyp verb command rebuild []
gyp verb command clean []
gyp verb clean removing "build" directory
gyp verb command configure []
gyp verb check python checking for Python executable "python2" in the PATH
gyp verb `which` failed Error: not found: python2
gyp verb `which` failed     at getNotFoundError (X:\IONIC\lab4a\ionicLab4\node_modules\which\which.js:13:12)
gyp verb `which` failed     at F (X:\IONIC\lab4a\ionicLab4\node_modules\which\which.js:68:19)
gyp verb `which` failed     at E (X:\IONIC\lab4a\ionicLab4\node_modules\which\which.js:80:29)
gyp verb `which` failed     at X:\IONIC\lab4a\ionicLab4\node_modules\which\which.js:89:16
gyp verb `which` failed     at X:\IONIC\lab4a\ionicLab4\node_modules\isexe\index.js:42:5
```

ERROR 2 (Possible) – if we get an error message with ‘vendor’ and node-sass. Go into the node_modules folder, into node-sass and create a folder called vendor



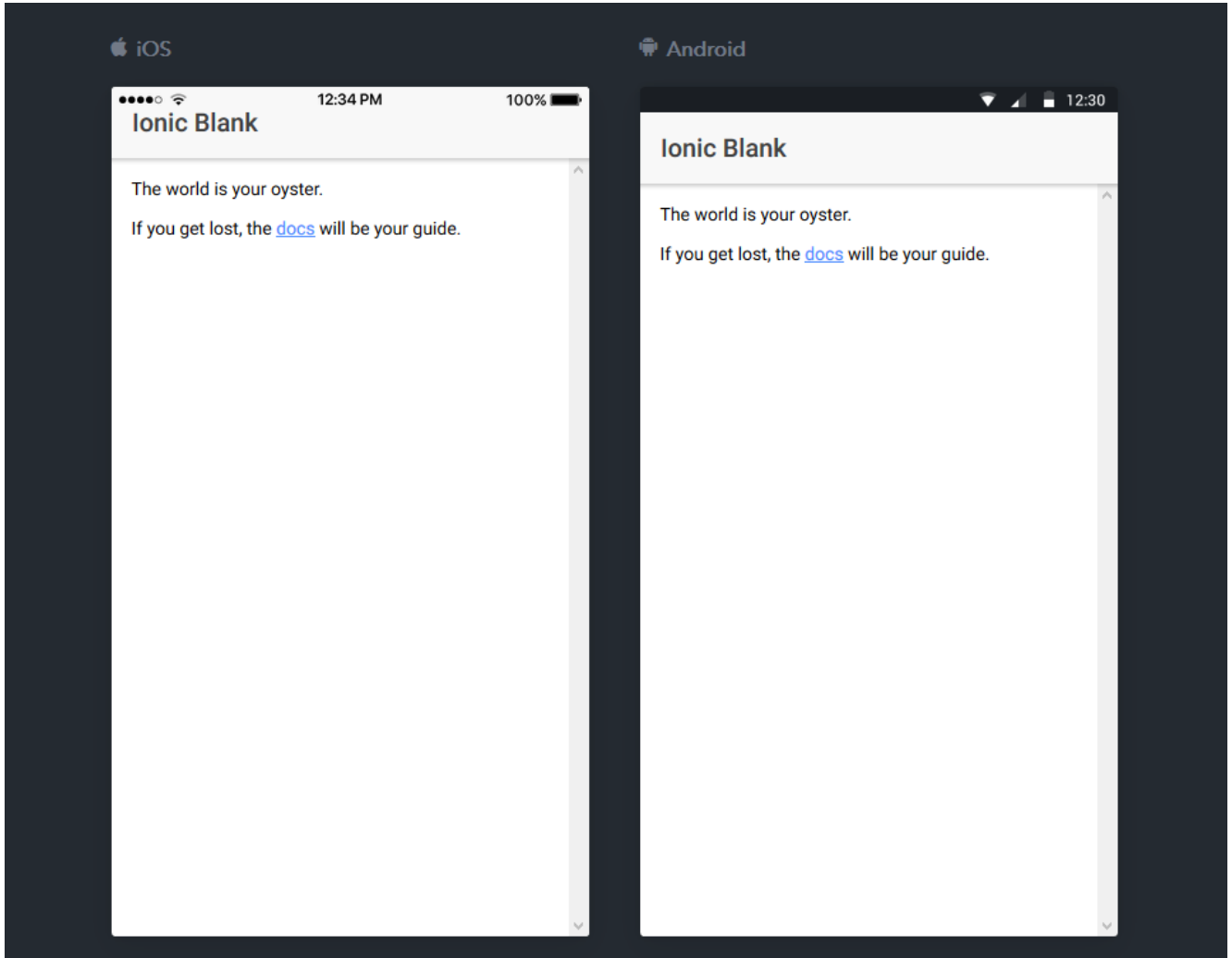
SERVE – when you try to serve the application you'll be asked to install ionic/lab – type Y

```
[app-scripts] [14:47:30]  transpile started ...
[app-scripts] [14:47:40]  transpile finished in 9.81 s
[app-scripts] [14:47:40]  preprocess started ...
[app-scripts] [14:47:40]  preprocess finished in less than 1 ms
[app-scripts] [14:47:40]  webpack started ...
[app-scripts] [14:47:40]  copy finished in 10.61 s
[app-scripts] [14:47:44]  webpack finished in 4.52 s
[app-scripts] [14:47:44]  sass started ...
[app-scripts] [14:47:46]  sass finished in 1.96 s
[app-scripts] [14:47:46]  postprocess started ...
[app-scripts] [14:47:46]  postprocess finished in 16 ms
[app-scripts] [14:47:46]  lint started ...
[app-scripts] [14:47:46]  build dev finished in 16.56 s
[app-scripts] [14:47:47]  watch ready in 16.86 s
> ionic-lab http://localhost:8100 --host localhost --port 8200 --app-name ionicLab4 --app-vers
[lab] 'ionic-lab' is not recognized as an internal or external command,
[lab] operable program or batch file.

[INFO] Looks like @ionic/lab isn't installed in this project.

       This package is required for this command to work properly.

? Install @ionic/lab? (Y/n)
```

If the error messages with node-sass persist then try these two commands

- **COMMAND 1 npm rebuild node-sass –force**
- **That’s actually two minus symbols then force**
- **When that’s finished**
- **Type**
- **COMMAND 2 npm install**

Your browser should open a display like this – if not you can manually open the URL using the localhost address (as we did for Angular)



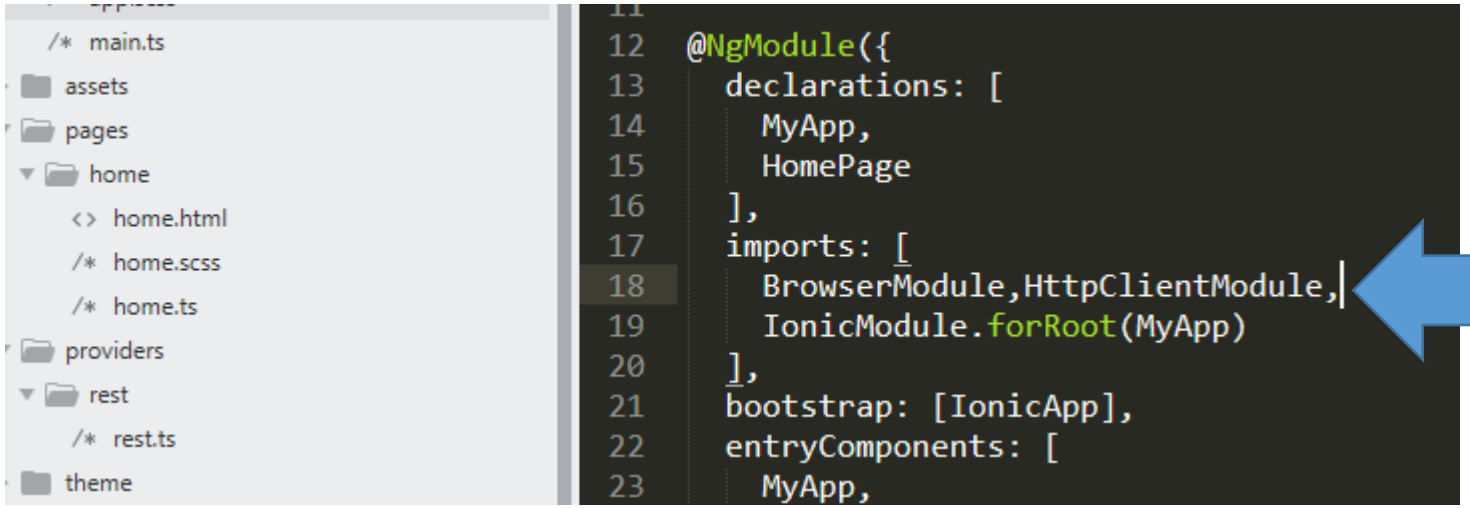
STEP 1: We need to include the HttpClientModule in our application

- This is necessary if we want to access resources on the Internet.
- We use HTTP and the HttpClientModule is a module/library in Angular which can help us simplify how we make ‘calls’ to things like APIs.
- So let’s get that installed or included first.

STEP 1: Include the HttpClientModule in app.module.ts

Add the following line to the top of the file (app.module.ts). Be very careful of spelling. DO NOT copy and paste from PDF. Every application using the HttpClientModule will need to import this library/module.

```
import { HttpClientModule } from '@angular/common/http';
```



We must also add this to our list of IMPORTS in **app.module.ts** (Careful of spelling and commas)

STEP 2: Make some changes to home.html – just write some content between the <ion-content> tags

X:\IONIC\lab4a\ionicLab4\src\pages\home\home.html (ionicLab4) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

- ionicLab4
  - .sourcemaps
  - node_modules
  - resources
  - src
    - app
      - app.component.ts
      - app.html
      - app.module.ts
      - app.scss
      - main.ts
    - assets
    - pages
      - home
        - home.html
        - home.scss
        - home.ts
    - theme
    - index.html

home.html

```
<ion-header>
  <ion-navbar>
    <ion-title>
      Lab 4 CS385 - Ionic
    </ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding>
  The world is your oyster.
  <p>
    If you get lost, the <a href="http://ionicframework.com/
    docs/v2">docs</a> will be your guide.
  </p>
</ion-content>
```

CHANGE THIS

Don’t delete the <ion-content tags>

STEP 2: Save your file and you should see the Ionic Application in the Labs browser update!

Pay attention to closing tags!

X:\IONIC\lab4a\ionicLab4\src\pages\home\home.html (ionicLab4) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

- ionicLab4
  - .sourcemaps
  - node_modules
  - resources
  - src
    - app
      - app.component.ts
      - app.html
      - app.module.ts
      - app.scss
      - main.ts
    - assets
    - pages
      - home
        - home.html
        - home.scss
        - home.ts
    - theme
    - index.html

home.html

```
<ion-header>
  <ion-navbar>
    <ion-title>
      Lab 4 CS385 - Ionic
    </ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding>
  The world is your oyster.
  <p>
    If you get lost, the <a href="http://ionicframework.com/
    docs/v2">docs</a> will be your guide.
  </p>
</ion-content>
```

CHANGE THIS

Don’t delete the <ion-content tags>

STEP 3 – Generate a PROVIDER. This will allow us to isolate code related to actually accessing external HTTP resources in one Typescript file

- As explained in the lecture (9) – the provider is the class where we access the external data resource(s).
- This PROVIDER creates a PROMISE object which essentially contains the data downloaded from the external resource.
- Inside the PROVIDER Typescript file we specify the URL or address of where the resource is on the Internet.
- **For these examples we make the valid assumption that these external resources will generate JSON for us to consume. Later in CS385 we'll deal with situations where there might be errors or EMPTY JSON responses.**

STEP 3: Generate a provider – which will be the way that we will access the REST API

Type this command in the Node JS command prompt

ionic

```
X:\IONIC\lab4a\ionicLab4>ionic generate provider rest
```

Node.js command prompt

```
X:\IONIC\lab4a\ionicLab4>ionic generate provider rest
[OK] Generated a provider named rest!

X:\IONIC\lab4a\ionicLab4>
```

Our PROVIDER is called rest and notice that Ionic/Angular has created the boilerplate code for us

X:\IONIC\lab4a\ionicLab4\src\providers\rest\rest.ts (ionicLab4) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

- ionicLab4
  - .sourcemaps
  - node_modules
  - resources
  - src
    - app
      - app.component.ts
      - app.html
      - app.module.ts
    - providers
      - rest
        - rest.ts

home.html | rest.ts

```
1  import { HttpClient } from '@angular/common/http';
2  import { Injectable } from '@angular/core';
3
4  /*
5    Generated class for the RestProvider provider.
6
7    See https://angular.io/guide/dependency-injection fo
8    and Angular DI.
9  */
10 @Injectable()
11 export class RestProvider {
12
13   constructor(public http: HttpClient) {
14     console.log('Hello RestProvider Provider');
15   }
16
17 }
18
```

Step 3 – students using their own computers may not be able to run the generate provider command

- This might be due to having Ionic 4 installed (newest version)
- <u>THIS ONLY APPLIES TO STUDENTS USING THEIR OWN COMPUTERS – AND THIS PROBLEM MIGHT NOT EVEN OCCUR.</u>

- The alternative command is **ionic generate service rest**

STEP 4: We need to tell app.module.ts that we are using a provider called RestProvider from rest.ts

app.module.ts

```
1  import { BrowserModule } from '@angular/platform-browser';
2  import { ErrorHandler, NgModule } from '@angular/core';
3  import { IonicApp, IonicErrorHandler, IonicModule } from 'ionic-angular
4  import { SplashScreen } from '@ionic-native/splash-screen';
5  import { StatusBar } from '@ionic-native/status-bar';
6  import { HttpClientModule } from '@angular/common/http';
7
8  import { MyApp } from './app.component';
9  import { HomePage } from '../pages/home/home';
10 import { RestProvider } from '../providers/rest/rest';
11
12 @NgModule({
13   declarations: [
14     MyApp,
15     HomePage
16   ],
17   imports: [
18     BrowserModule,HttpClientModule,
19     IonicModule.forRoot(MyApp)
20   ],
21   bootstrap: [IonicApp],
22   entryComponents: [
23     MyApp,
24     HomePage
25   ],
26   providers: [
27     StatusBar,
28     SplashScreen,
29     {provide: ErrorH                    seClass: IonicErrorHandler},
30     RestProvider
31   ]
32 })
33 export class AppModule {}
```

Inserted automatically

```
10 @Injectable()
11 export class RestProvider {
12
```

Include RestProvider as a provider in the @NgModule section of the app.module.ts. Be careful of spelling and commas. The name RestProvider comes from our Class name in rest.ts

Step 5. Up to this point we have really just indicated in the application that we will be doing some work with the HTTPClient.

- Now we need to make some changes in rest.ts to indicate a number of important aspects of our code.
  - (1) We need to indicate the URL of the location on the Internet from where we are accessing the JSON data
  - (2) We need to include our code for accessing this URL and creating the PROMISE
- Remember the PROMISE will be used in other parts of our application (it's just an object which contains data – for our purposes)

Step 5A: Add the OUR_REST_API_URL variable to rest.ts. This is in the Moodle File for today

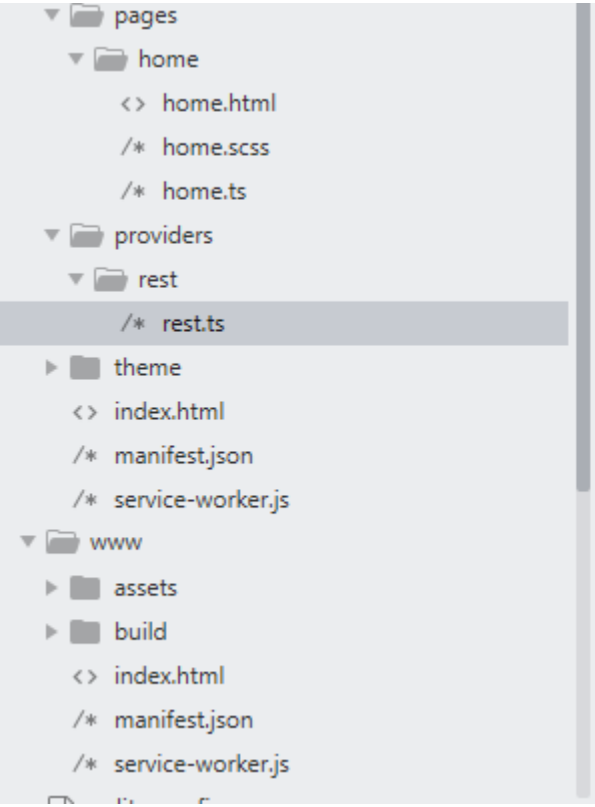
- Simply place this variable or property declaration just after the class opening and before the constructor.
- As always be careful of spelling and of semi colons etc.
- SAVE the file.
- https://www.cs.nuim.ie/~pmooney/api/api.php

```
10  @Injectable()
11  export class RestProvider {
12
13    OUR_REST_API_URL =  'https://api.iextrading.com/1.0/tops';
14
15    constructor(public http: HttpClient) {
16      console.log('Hello RestProvider Provider');
17    }
18
```

PLEASE NOTE THE CHANGE OF URL – THERE IS NOW A DIFFERENT URL for the API

- Please use this one
- https://www.cs.nuim.ie/~pmooney/api/api.php

STEP 5A: Add in the PROMISE code (after the constructor) from the Moodle File for today's lecture. This is rest.ts. Then SAVE




```
  constructor(public http: HttpClient) {
    console.log('Hello RestProvider Provider');
  }

/*
This is where our application accesses the external API for our data needs.
We use a promise.
We use the console.log to log any issues that occur.
*/
  getDataFromAPIViaPromise() {
  return new Promise(resolve => {
    this.http.get(this.OUR_REST_API_URL).subscribe(data => {
      resolve(data);
      console.log("Subscribed to the Data Promise");
    }, err => {
      console.log(err);
    });
  });
}
```

Check back to Lecture 9 for an explanation of this Promise code

We are now ready to actually use our RestProvider to download data from the URL and use it in our application.

- So our target for the data we download will be displaying the data in home.html (the template) and we will have the logic/code in the Typescript file (home.ts).
- This is exactly the same concept as we had with Angular.
- **Our PROMISE will deliver us an object which will be handled in the home.ts file and then the home.html (template) will render or display it on our application.**

What does our JSON data look like?

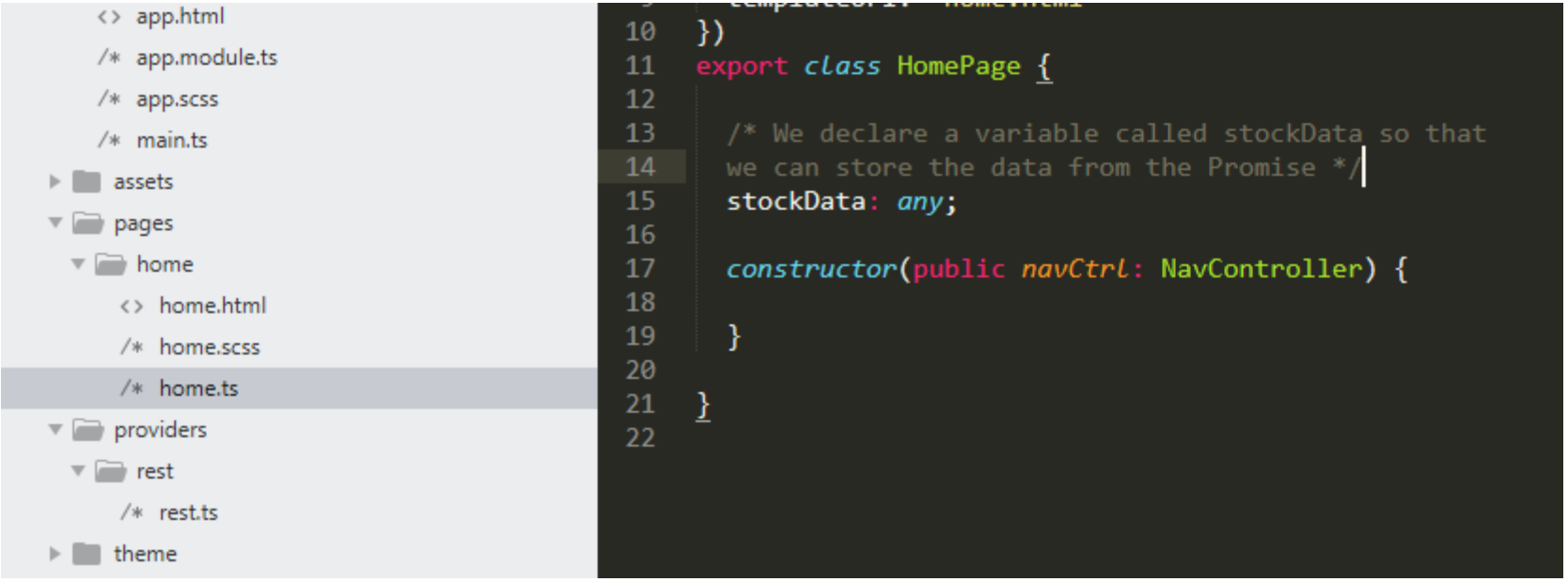
- For this example we are accessing some market trading data.
- This is what the JSON data looks like.
- We have covered JSON data objects in the Lecture
- So in **home.ts** we will create a property/variable called stockData to hold all of this data (from the Promise)


```
[
  {
    "symbol": "KWR",
    "sector": "materials",
    "securityType": "commonstock",
    "bidPrice": 0,
    "bidSize": 0,
    "askPrice": 0,
    "askSize": 0,
    "lastUpdated": 1540380631648,
    "lastSalePrice": 0,
    "lastSaleSize": 0,
    "lastSaleTime": 0,
    "volume": 0,
    "marketPercent": 0
  },
  {
    "symbol": "MHNC",
    "sector": "n/a",
    "securityType": "misc",
    "bidPrice": 0,
    "bidSize": 0,
    "askPrice": 0,
    "askSize": 0,
    "lastUpdated": 1540380631648,
    "lastSalePrice": 0,
    "lastSaleSize": 0,
    "lastSaleTime": 0,
    "volume": 0,
    "marketPercent": 0
  },
```

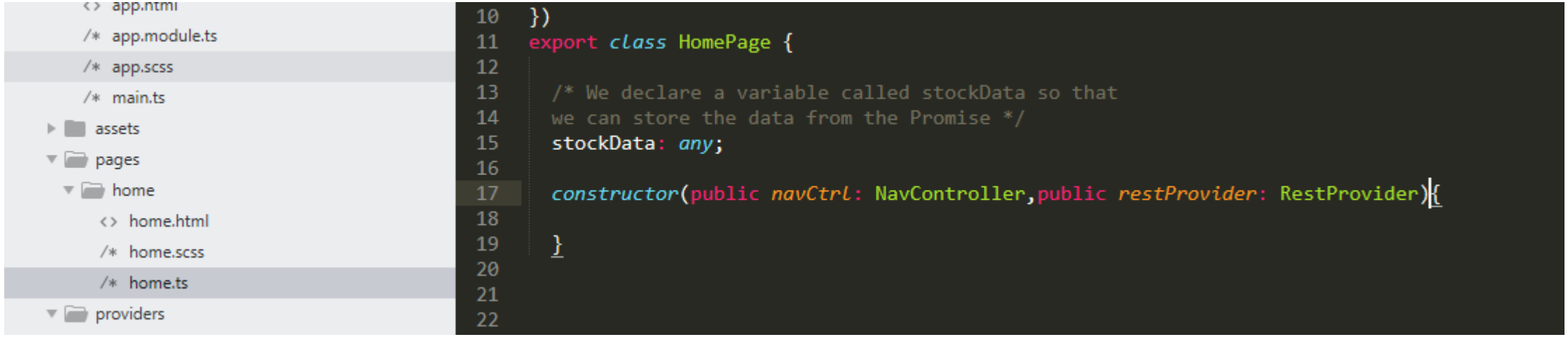
Step 6: So our provider is setup. Now we want to display the Stock data in our home page

- In our **home.ts** we must import our PROVIDER (check the Moodle file for the line below). It must be pasted into **home.ts**
- **import { RestProvider } from '../../providers/rest/rest';**
- Save the file. Do not copy and paste from this PDF.
- As the RestProvider is a class this will give us (home.ts) access to the method **`getDataFromAPIViaPromise()`** within the RestProvider class in rest.ts

Step 6A: Create a new property/variable in home.ts to hold the data from our Promise



Step 6a: In home.ts we must change the Constructor parameters to prepare for a RestProvider object



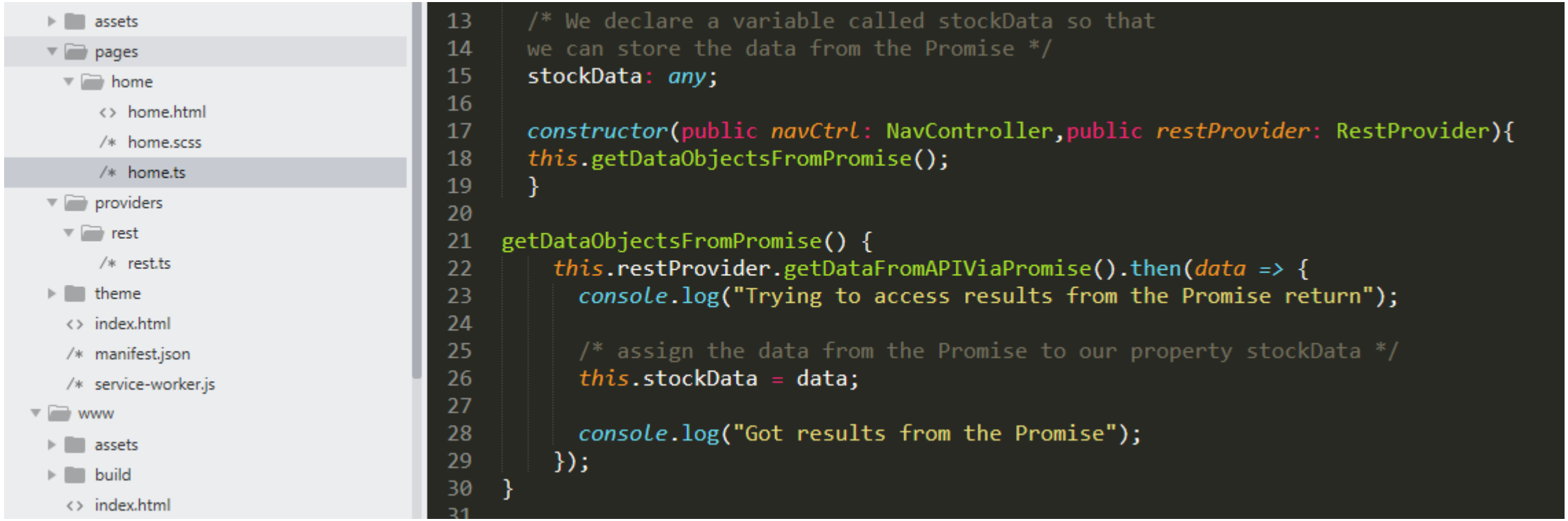
This is the new parameters in the constructor in home.ts.

The constructor makes a public object from the RestProvider

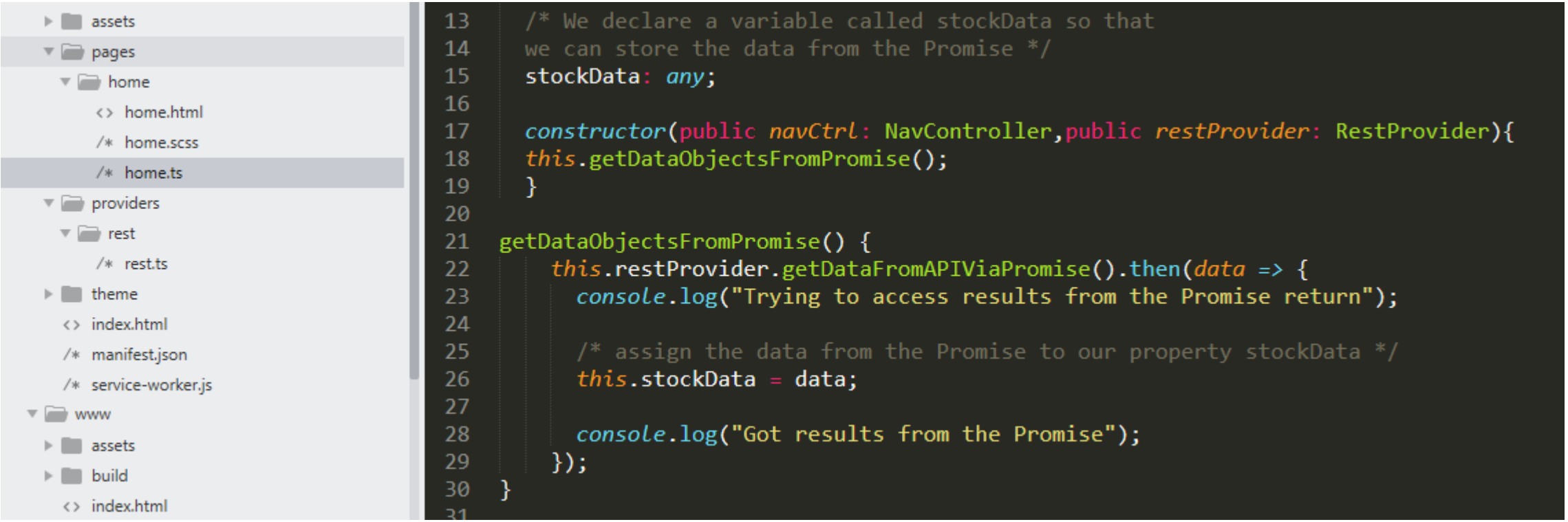
Be careful that you don't delete any curly brackets. Be careful of commas. Be careful of spelling

SAVE the changes.

Step 6b: Finally, in home.ts we need to have a function or method to assign stockData the data in the promise. We create our own function (copy from Moodle File)



Step 6b: It is CRUCIAL to REALLY UNDERSTAND what is happening here in this code.



Step 6b: It is CRUCIAL to REALLY UNDERSTAND what is happening here in this code.

```
13    /* We declare a variable called stockData so that
14    we can store the data from the Promise */
15    stockData: any;
16
17    constructor(public navCtrl: NavController,public restProvider: RestProvider){
18    this.getDataObjectsFromPromise();
19    }
20
21  getDataObjectsFromPromise() {
22      this.restProvider.getDataFromAPIViaPromise().then(data => {
23        console.log("Trying to access results from the Promise return");
24
25        /* assign the data from the Promise to our property stockData */
26        this.stockData = data;
27
28        console.log("Got results from the Promise");
29      });
30  }
```

1. **The constructor is called when the home.html page is invoked (hence calling home.ts).**
2. **The constructor calls the getDataObjectsFromPromise() method.**
3. **This method then calls the getDataFromAPIViaPromise method (in rest.ts).**
4. **This method returns the data from the API call (JSON) and stores it in a variable called data. This is a JSON object.**
5. **The getDataObjectsFromPromise() then assigns the stockData property with the variable called data. So stockData now holds all of the JSON we seen earlier.**

Step 6c: Now stockData property contains all of our JSON. Now we need to prepare home.html to display the data.

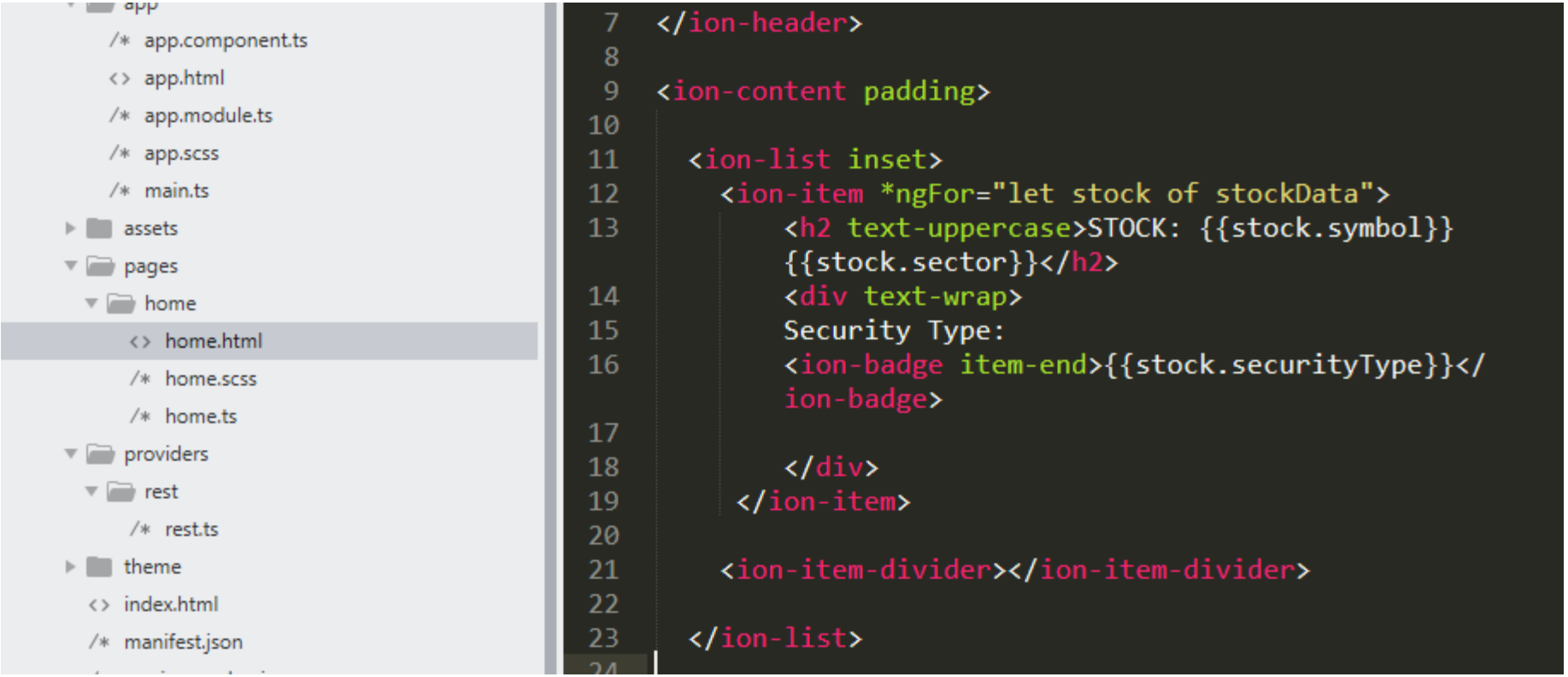
https://api.iextrading.com/1.0/tops

JSON Raw Data Headers

Save Copy Pretty Print


```
[
  {
    "symbol": "KWR",
    "sector": "materials",
    "securityType": "commonstock",
    "bidPrice": 0,
    "bidSize": 0,
    "askPrice": 186.7,
    "askSize": 100,
    "lastUpdated": 1540301466836,
    "lastSalePrice": 0,
    "lastSaleSize": 0,
    "lastSaleTime": 0,
    "volume": 0,
    "marketPercent": 0
  },
  {
    "symbol": "MHNC",
    "sector": "n/a",
    "securityType": "misc",
    "bidPrice": 0,
    "bidSize": 0,
    "askPrice": 0,
    "askSize": 0,
    "lastUpdated": 1540301543901,
    "lastSalePrice": 0,
    "lastSaleSize": 0,
    "lastSaleTime": 0,
    "volume": 0,
    "marketPercent": 0
  },
```

Step 7: Copy the home.html code from the Moodle file into home.html – save the changes.



STEP 7: It is CRUCIAL to Understand the relationship between the JSON data and the {{ }} outputs using stockData variable

We use *ngFor again to iterate over the stockData variable (essentially an array of JSON objects)

https://api.iextrading.com/1.0/tops

JSON | Raw Data | Headers

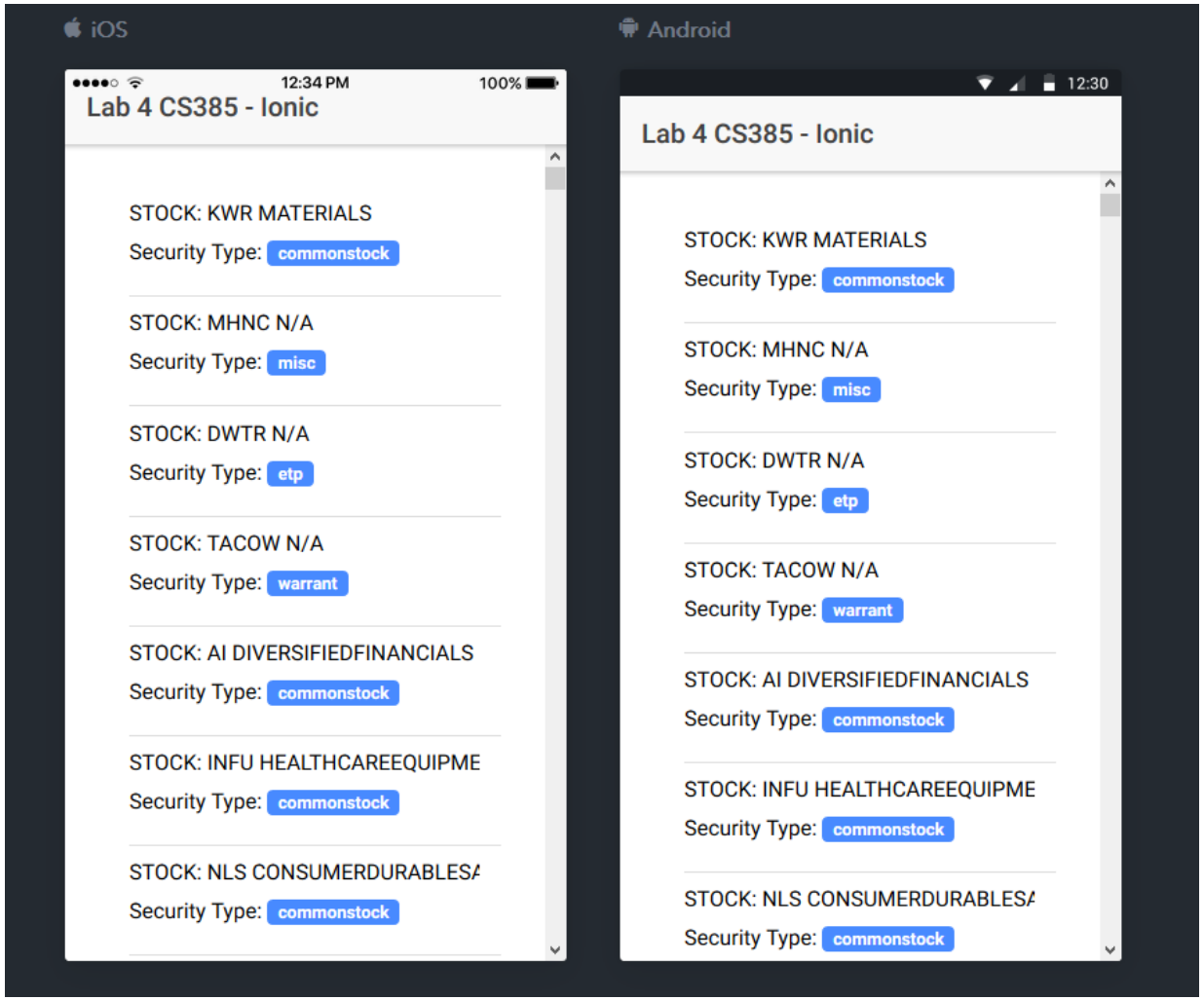
Save | Copy | Pretty Print


```
[
  {
    "symbol": "XXIP",
    "sector": "materials",
    "securityType": "commonstock",
    "bidPrice": 0,
    "bidSize": 0,
    "askPrice": 186.7,
    "askSize": 100,
    "lastUpdated": 1540301466836,
    "lastSalePrice": 0,
    "lastSaleSize": 0,
    "lastSaleTime": 0,
    "volume": 0,
    "marketPercent": 0
  },
  {
    "symbol": "MHNC",
    "sector": "n/a",
    "securityType": "misc",
    "bidPrice": 0,
    "bidSize": 0,
    "askPrice": 0,
    "askSize": 0,
    "lastUpdated": 1540301543901,
    "lastSalePrice": 0,
    "lastSaleSize": 0,
    "lastSaleTime": 0,
    "volume": 0,
    "marketPercent": 0
  },
```


- app
  - app.component.ts
  - app.html
  - app.module.ts
  - app.scss
  - main.ts
- assets
- pages
  - home
    - home.html
    - home.scss
    - home.ts
- providers
  - rest
    - rest.ts
- theme
- index.html
- manifest.json

```
7   </ion-header>
8
9   <ion-content padding>
10
11    <ion-list inset>
12      <ion-item *ngFor="let stock of stockData">
13        <h2 text-uppercase>{{stock.symbol}}
          {{stock.sector}}</h2>
14        <div text-wrap>
15        Security Type:
16        <ion-badge item end>{{stock.securityType}}</
          ion-badge>
17
18        </div>
19      </ion-item>
20
21      <ion-item-divider></ion-item-divider>
22
23    </ion-list>
```

Step 7 – if everything has worked then you should see the outputs on **home.html** (from the API call) on our application



PROBLEM – What if you are not seeing ANY of the stock data output? What if the error is displayed indicating a Null Pointer or Null Reference to a REST PROVIDER

- Go to your NodeJS command prompt where ionic serve –lab is running.
- Then type CTRL-C together
- Then type Y for yes.
- Then run the command ionic serve –lab again (minus minus)
- This will ensure that the serve will rebuild the entire project again

PAUSE FOR REVIEW

- At this point we can see that we can VERY EASILY reuse this code for different API URLs (provided they serve us JSON data).
- We can easily change the home.html to display the output differently (and more stylishly).
- However – there is really no interaction here.
- **Our next step is to allow users to CLICK on one of the items returned from the API Call and then display this item/object on a new page.**
- **This will illustrate how we move to other pages in Ionic AND pass data objects between pages in an Ionic application.**

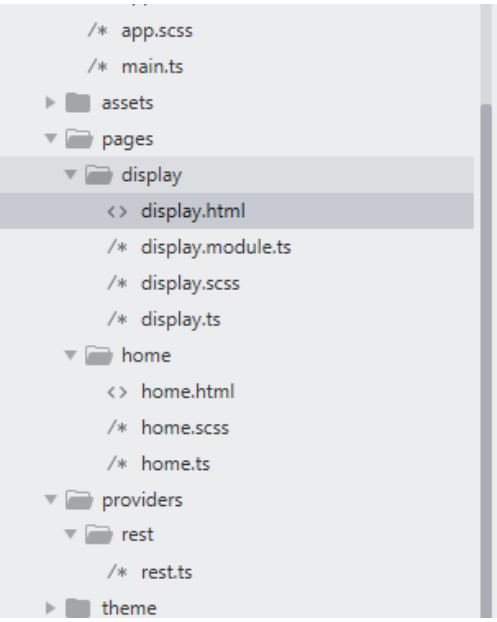
Creating an interaction in our application AND learning about the NavController

Step 8 – Let's create a special PAGE where we will display the full details of a JSON object

- As before – go to the Node JS Command Prompt for the application and type the command **`ionic generate page display`**

- Our page will be called **display.html** and our typescript file will be **display.ts**

```
The system cannot find the path specified.

X:\IONIC\lab4>cd ..

X:\IONIC>cd lab4a

X:\IONIC\lab4a>cd ionicLab4

X:\IONIC\lab4a\ionicLab4>ionic generate page display
[OK] Generated a page named display!

X:\IONIC\lab4a\ionicLab4>
```



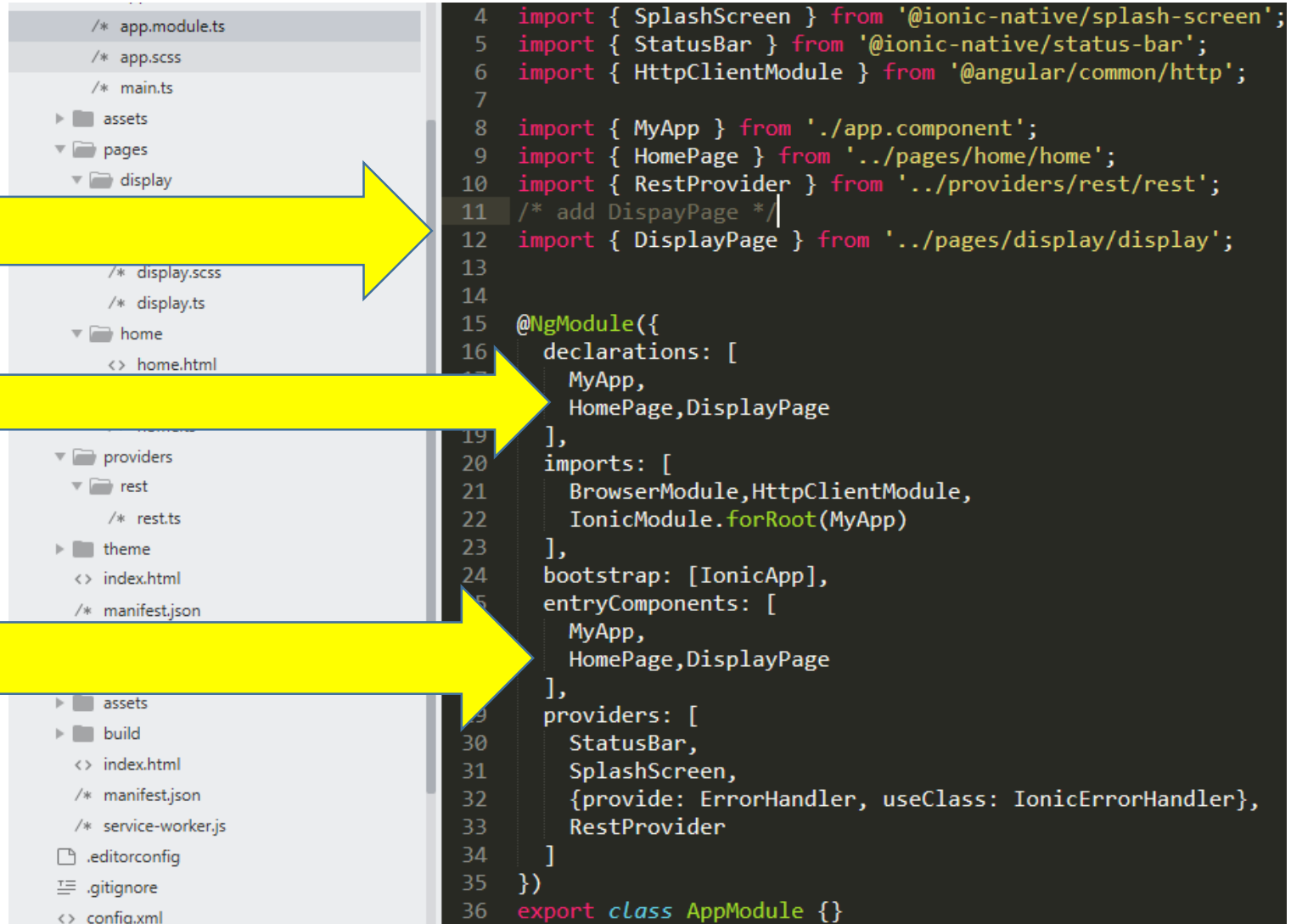

```
  Generated template for the Display

  See http://ionicframework.com/docs
  Ionic pages and navigation.
-->
<ion-header>

  <ion-navbar>
    <ion-title>display</ion-title>
  </ion-navbar>

</ion-header>


<ion-content padding>

</ion-content>

```

Step 9 – we must tell **app.module.ts** that we have a page called **display**

We must include THREE pieces of information in **app.module.ts**

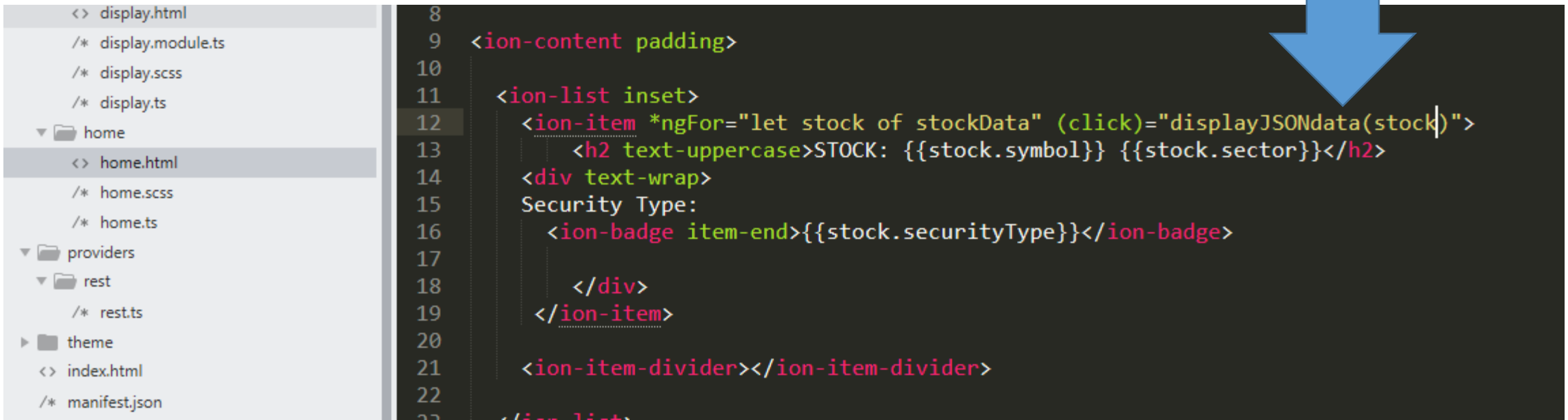
Be careful of spelling and of commas.

SAVE file after the changes.



Step 10 – now we need to prepare home.ts and home.html so that we capture the object which the user clicks on. We need to store that object

- Firstly we need to make a CLICK event on the objects on home.html
- As below – add a click event (you need to type this in) – which will pass the variable/property stock to a method called **displayJSONData()** in home.ts



Step 10a. Provide the code for displayJSONData in home.ts

- Declare a variable called selectedObject (as below) just after the stockData variable. You don't need to include comments. This will be the object that the user clicks on
- Then COPY the displayJSONData code from the Moodle File

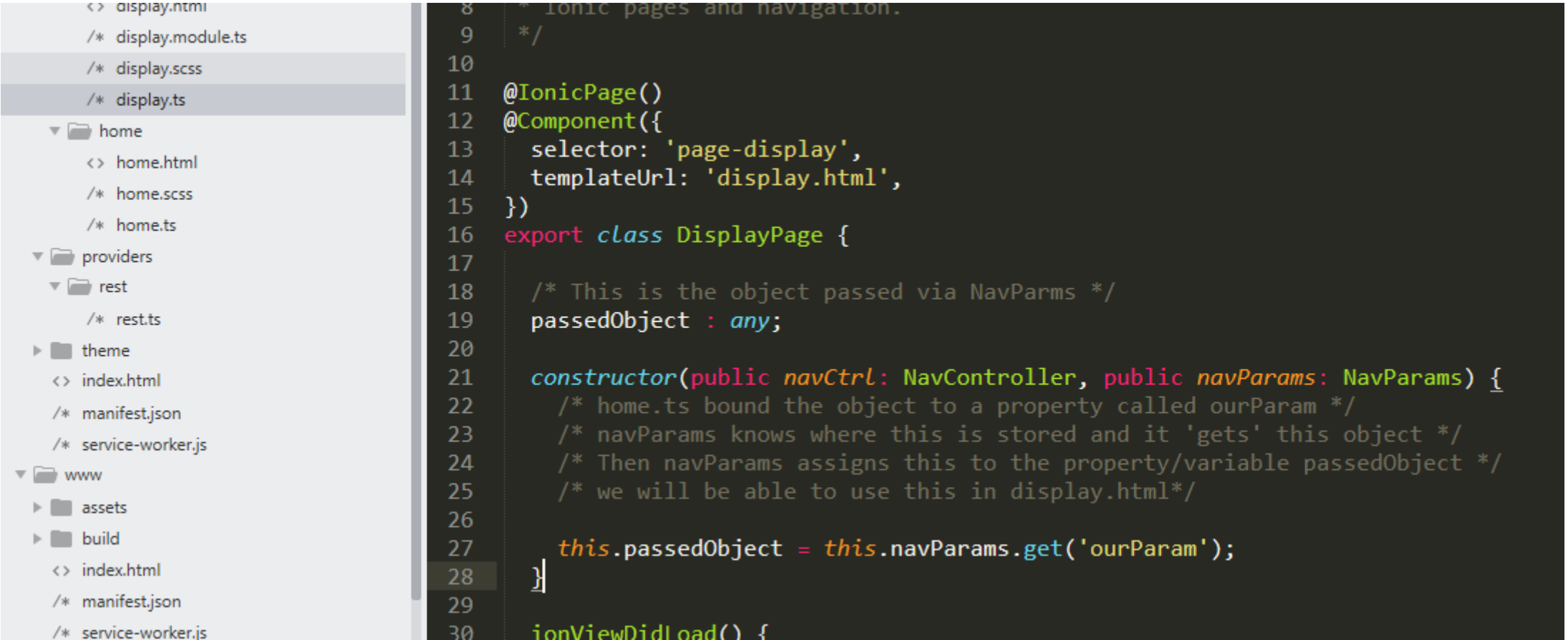
```
11  export class HomePage {
12
13    /* We declare a variable called stockData so that
14    we can store the data from the Promise */
15    stockData: any;
16
17    selectedObject: any;  // this will hold the clicked object in home.ts
18    // so this will be available for use by other ts files and pages.
19
20    constructor(public navCtrl: NavController,public restProvider: RestProvider){
21    this.getDataObjectsFromPromise();
22    }
23
24    displayJSONdata(clickedObject: any): void {
25      this.selectedObject = clickedObject;
26      console.log("Assigned the JSON object in the Click Event from home.html");
27
28    }
29
```

Step 10 – at this point home.ts knows the object which has been clicked on.

- Now we have to consider the **NavControllers** which we seen in the lectures.
- We want to be able to pass this clicked object in **home.ts** to our **details.ts** and details.html page.
- **This is DATA PASSING between Ionic Pages** (and is similar in the concept we had for Angular Components in Lab 3)
- This is a VERY IMPORTANT concept – as it will allow your application to move data between pages efficiently and effectively

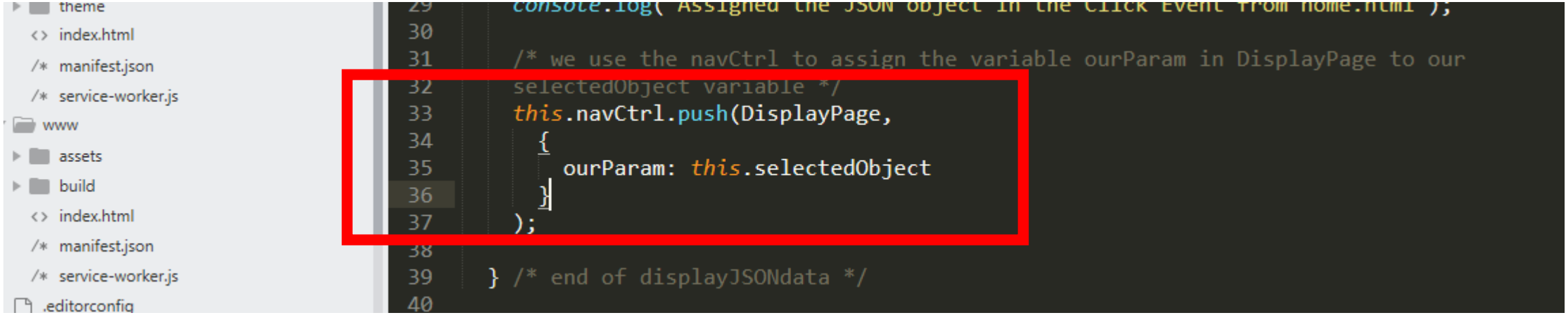
Step 11 – let's prepare the display.ts file

- Copy the code from the Moodle file and completely replace the constructor in **display.ts**



Step 12. We've created a NavParam so we must specify this in home.ts (it will use a NavParam) to pass the data to details.ts

- We make the change below in our **displayJSONdata** method.
- This is where the parameter passing happens



Step 12. We've created a NavParam so we must specify this in home.ts (it will use a NavParam) to pass the data to details.ts

- We also need to import the **DisplayPage** so that **home.ts** knows where the display page is.
- `import {DisplayPage} from "../display/display";`

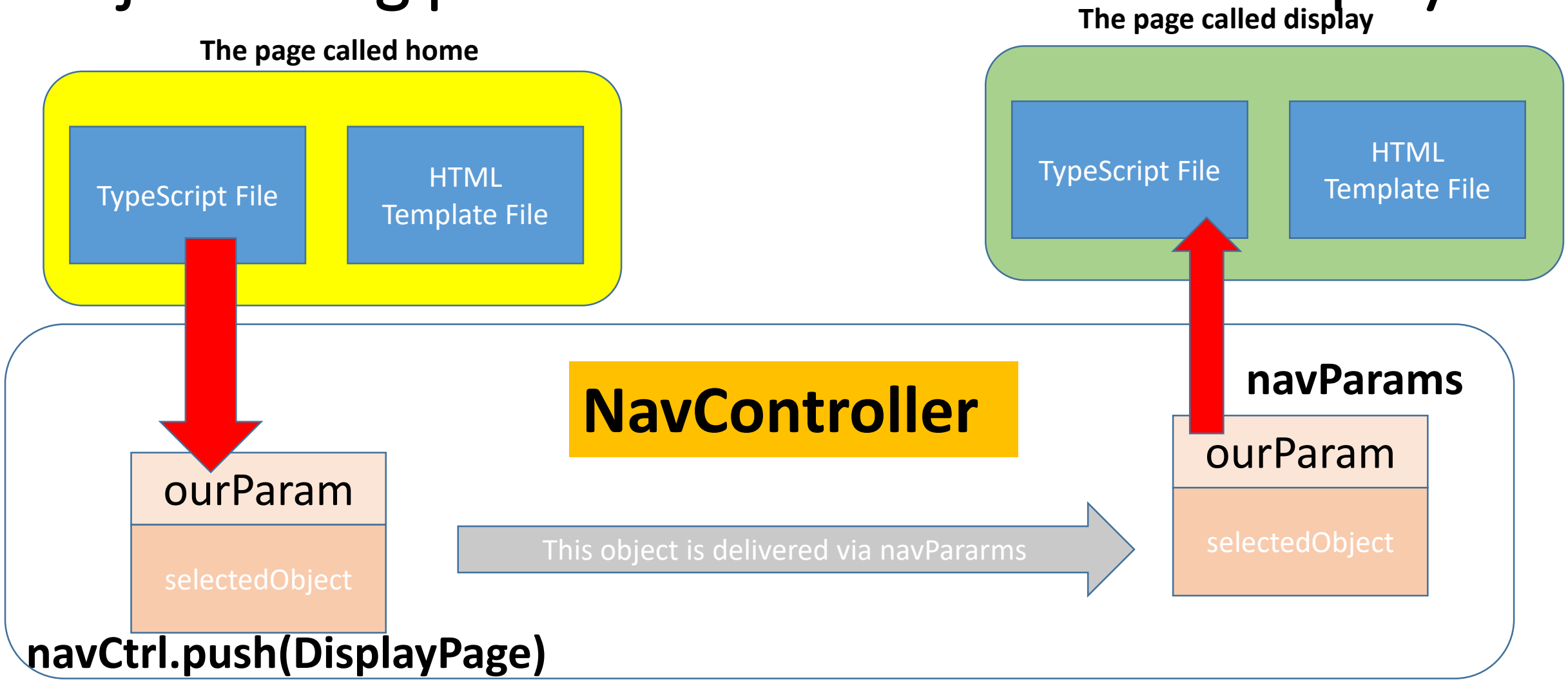
```
rest.ts   app.module.ts   home.html   home.ts
import { Component } from '@angular/core';
import { NavController } from 'ionic-angular';
import { RestProvider } from '../../providers/rest/rest';
import {DisplayPage} from "../display/display";

|
```

FINALLY ….. We just need to create our TEMPLATE for display.html

- Remember in **display.ts** we now have access to the variable/property containing the object which the user clicked on.
- It is called passedObject – so in **display.html** we will reference this using our {{}}
- CRUCIALLY – we need to know the JSON structure for our display.html

```
export class DisplayPage {

  /* This is the object passed via NavParms */
  passedObject : any;

  constructor(public navCtrl: NavController, public navParams: NavParams) {
    /* home.ts bound the object to a property called ourParam */
    /* navParams knows where this is stored and it 'gets' this object */
    /* Then navParams assigns this to the property/variable passedObject */
    /* we will be able to use this in display.html*/

    this.passedObject = this.navParams.get('ourParam');
  }
```

What's actually happening here? How is the object being passed between home and display?



Step 13 – COPY all of the contents for **display.html** from the Moodle File

- Replace the entire contents of display.html with the contents from Moodle.
- SAVE
- Then hopefully, we’ll see some interactivity on our application

The passedObject variable is just JSON and we can reference the attributes directly

```
  <ion-navbar>
    <ion-title>This is {{passedObject.symbol}}</ion-title>
  </ion-navbar>

</ion-header>


<ion-content padding>
    <div text-justify>
        <h1>You clicked on {{passedObject.symbol}}</h1>

        <p>The details below are DIRECTLY from the JSON object which you clicked
        on in the previous screen on home.html</p>

        <p><b>SECTOR:</b> {{passedObject.sector}}</p>
        <p><b>SECURITY TYPE:</b>  {{passedObject.securityType}}</p>
        <p><b>Last Updated (Timestamp):</b>  {{passedObject.lastUpdated}}</p>
        <p><b>Bid Price:</b>  {{passedObject.bidPrice}}</p>
        <p><b>Ask Price:</b>  {{passedObject.askPrice}}</p>
        <p><b>Volume: </b> {{passedObject.volume}}</p>
        <p><b>Last Sale Price:</b>  {{passedObject.lastSalePrice}}</p>
        <p><b>Last Sale Size: </b> {{passedObject.lastSaleSize}}</p>

    </div>

</ion-content>
```

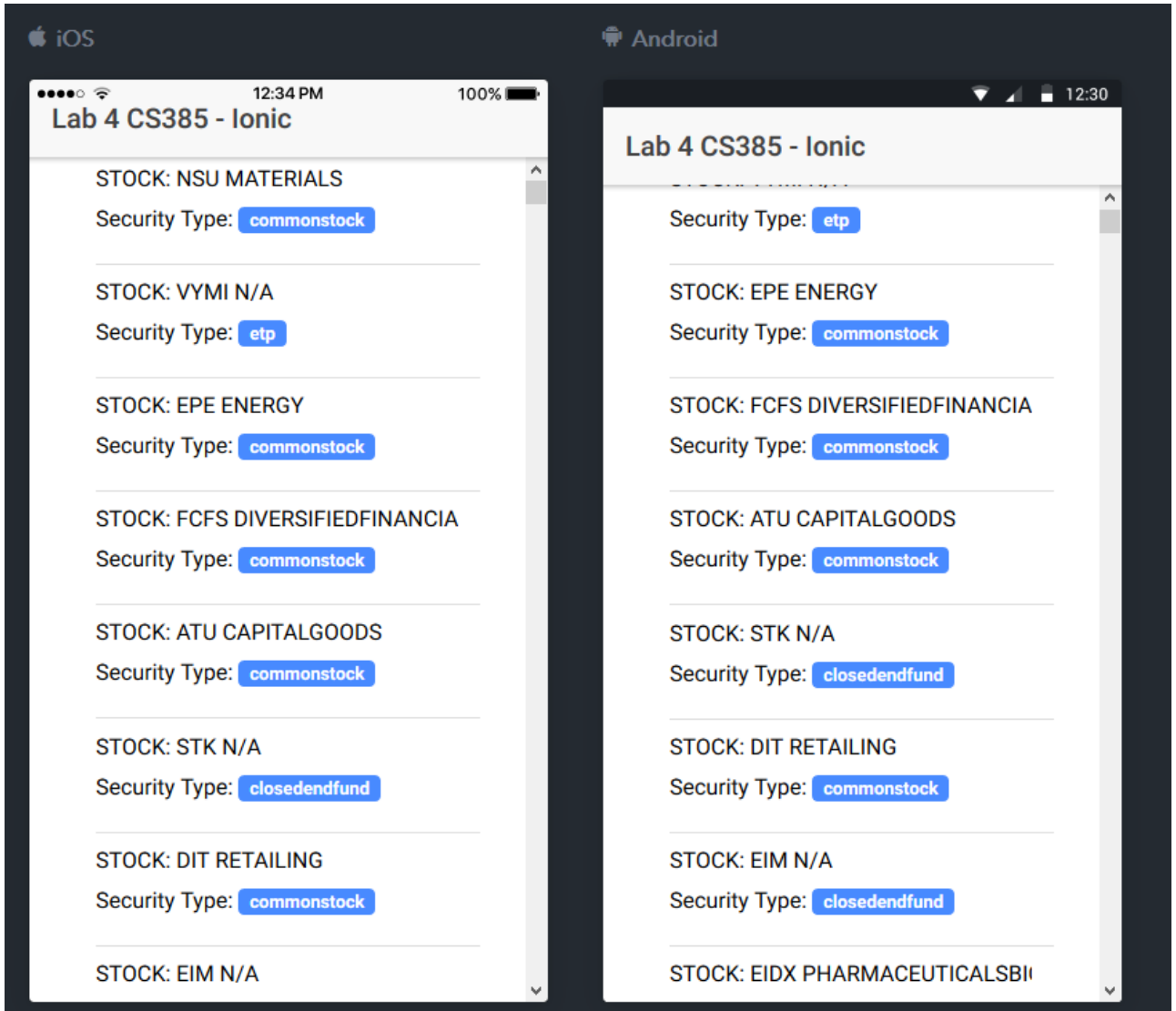

```
},
{
  "symbol": "AI",
  "sector": "diversifiedfinancials",
  "securityType": "commonstock",
  "bidPrice": 0,
  "bidSize": 0,
  "askPrice": 8.61,
  "askSize": 100,
  "lastUpdated": 1540390095731,
  "lastSalePrice": 8.59,
  "lastSaleSize": 100,
  "lastSaleTime": 1540389642379,
  "volume": 11763,
  "marketPercent": 0.07435
},
```

It is then our job to make this page look good using Ionic Components and HTML

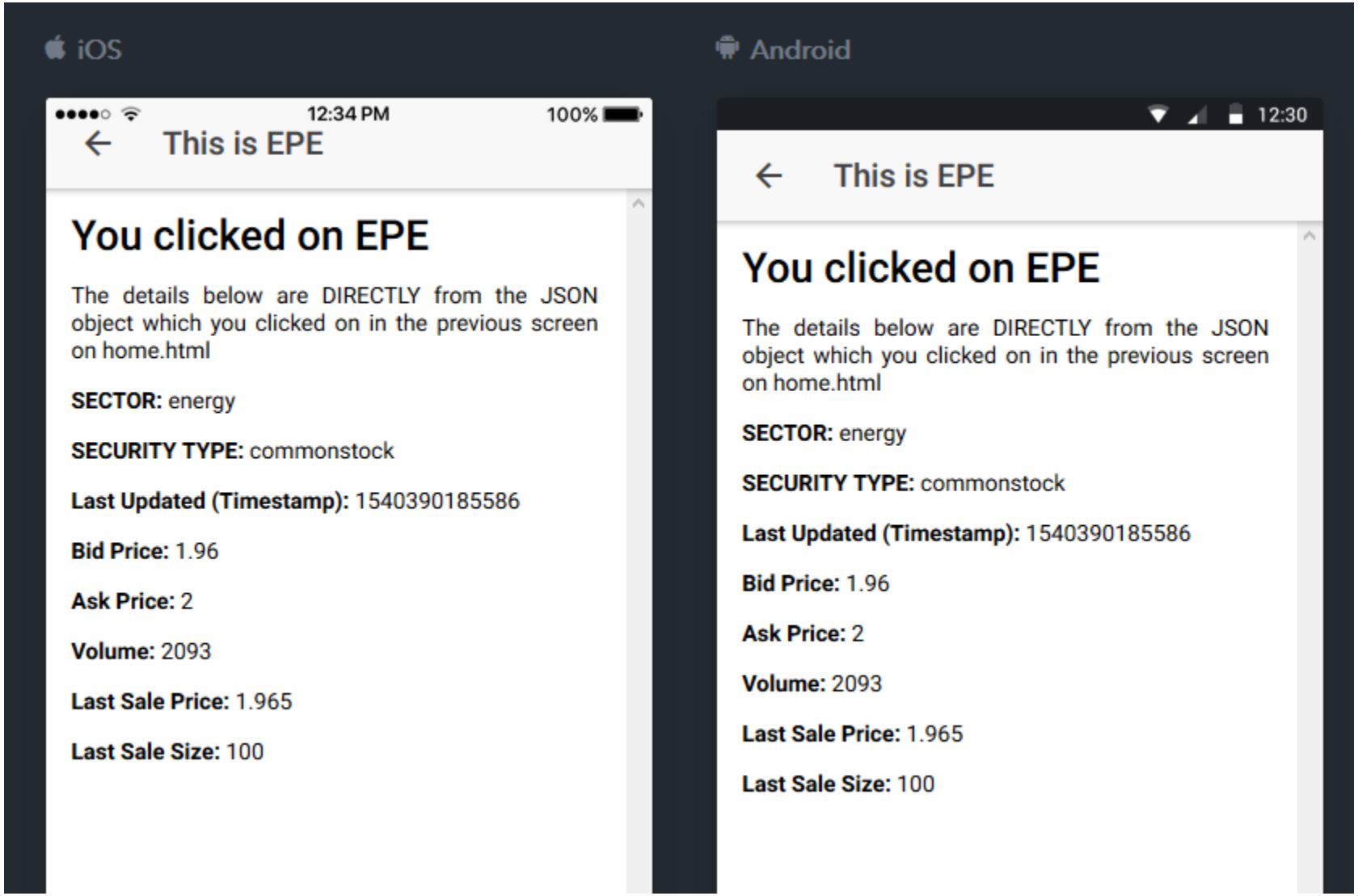
```
  <ion-navbar>
    <ion-title>This is {{passedObject.symbol}}</ion-title>
  </ion-navbar>

</ion-header>


<ion-content padding>
    <div text-justify>
        <h1>You clicked on {{passedObject.symbol}}</h1>

        <p>The details below are DIRECTLY from the JSON object which you clicked
        on in the previous screen on home.html</p>

        <p><b>SECTOR:</b> {{passedObject.sector}}</p>
        <p><b>SECURITY TYPE:</b>  {{passedObject.securityType}}</p>
        <p><b>Last Updated (Timestamp):</b>  {{passedObject.lastUpdated}}</p>
        <p><b>Bid Price:</b>  {{passedObject.bidPrice}}</p>
        <p><b>Ask Price:</b>  {{passedObject.askPrice}}</p>
        <p><b>Volume: </b> {{passedObject.volume}}</p>
        <p><b>Last Sale Price:</b>  {{passedObject.lastSalePrice}}</p>
        <p><b>Last Sale Size: </b> {{passedObject.lastSaleSize}}</p>

    </div>

</ion-content>
```


```
},
{
  "symbol": "AI",
  "sector": "diversifiedfinancials",
  "securityType": "commonstock",
  "bidPrice": 0,
  "bidSize": 0,
  "askPrice": 8.61,
  "askSize": 100,
  "lastUpdated": 1540390095731,
  "lastSalePrice": 8.59,
  "lastSaleSize": 100,
  "lastSaleTime": 1540389642379,
  "volume": 11763,
  "marketPercent": 0.07435
},
```

Step 13: Output – Click on EPE Energy



Step 13 – Output (NavCtrl)



GREAT – We’ll done

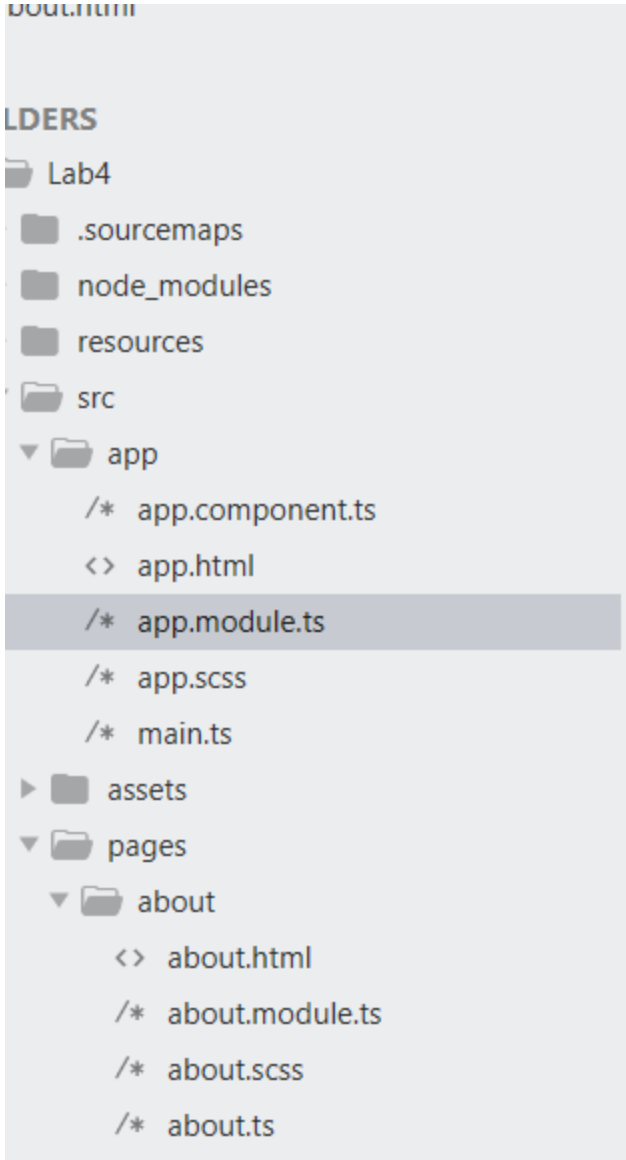
- We’ve got a basic but very useful mobile application fully setup.
- Can you see how you could change this code to use a different API?
- This code is VERY REUSABLE

PAGES – Let’s add another page and explore how to navigate around our application.

STEP 14: Let’s generate a page called about

- Remember how we generated the display page?
- Reuse this command to generate a page called about with Ionic

REMEMBER – Add the about page to app.module.ts




```
12  import { DisplayPage } from '../pages/display/display';
13  import { AboutPage } from '../pages/about/about';
14
15
16  @NgModule({
17    declarations: [
18      MyApp,
19      HomePage,DisplayPage,AboutPage
20    ],
21    imports: [
22      BrowserModule,HttpClientModule,
23      IonicModule.forRoot(MyApp)
24    ],
25    bootstrap: [IonicApp],
26    entryComponents: [
27      MyApp,
28      HomePage,DisplayPage,AboutPage
29    ],
30    providers: [
31      StatusBar,
```

Step 14 – what do we want to do?

- We want to put a button on the home.html page saying “Go to About”
- We want to put a button on the about.html page saying “Go to home”.
- Then we want to use the **NavController** to allow us to add navigation between these two pages.

Step 14a – copy from the Moodle File

- Go to the Moodle file and copy the button HTML code for “home.html” into the top of the home.html file
- Then copy the button HTML code for “about.html” into the top of the about.html file.

Home.html

```
9  <ion-content padding>
10 <ion-item>
11 <button ion-button round outline color="danger" (click)="goToAbout()">
   Go to About</button>
12 </ion-item>
13
14   <ion-list inset>
15     <ion-item *ngFor="let stock of stockData" (click)="displayJSONdata
       (stock)">
```

About.html

```
12
13 </ion-header>
14
15
16 <ion-content padding>
17     <ion-item>
18 <button ion-button round outline color="danger" (click)="goToHome()">
   Go to Home</button>
19 </ion-item>
20
21 </ion-content>
```

Step 14b. Now we have to go to our Typescript and supply the code to actually NAVIGATE to the page specified when the button is clicked

Step 14b – we have to ensure that home.html and about.html know where each other are – we need to import the pages into each Typescript file

- You need to type these yourself.

home.html | home.ts | app.module.ts | about.html | about.ts

```
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams } from 'ionic-angular';
import {HomePage} from "../home/home";

/**
```

For about.ts

```
import { NavController } from 'ionic-angular';
import { RestProvider } from '../../providers/rest/rest';
import {DisplayPage} from "../display/display";
import {AboutPage} from "../about/about";
```

For home.ts

Step 14B – copy the Typescript function goToHome() into the about.ts file

- Paste it into the file just before the last closing curly bracket.

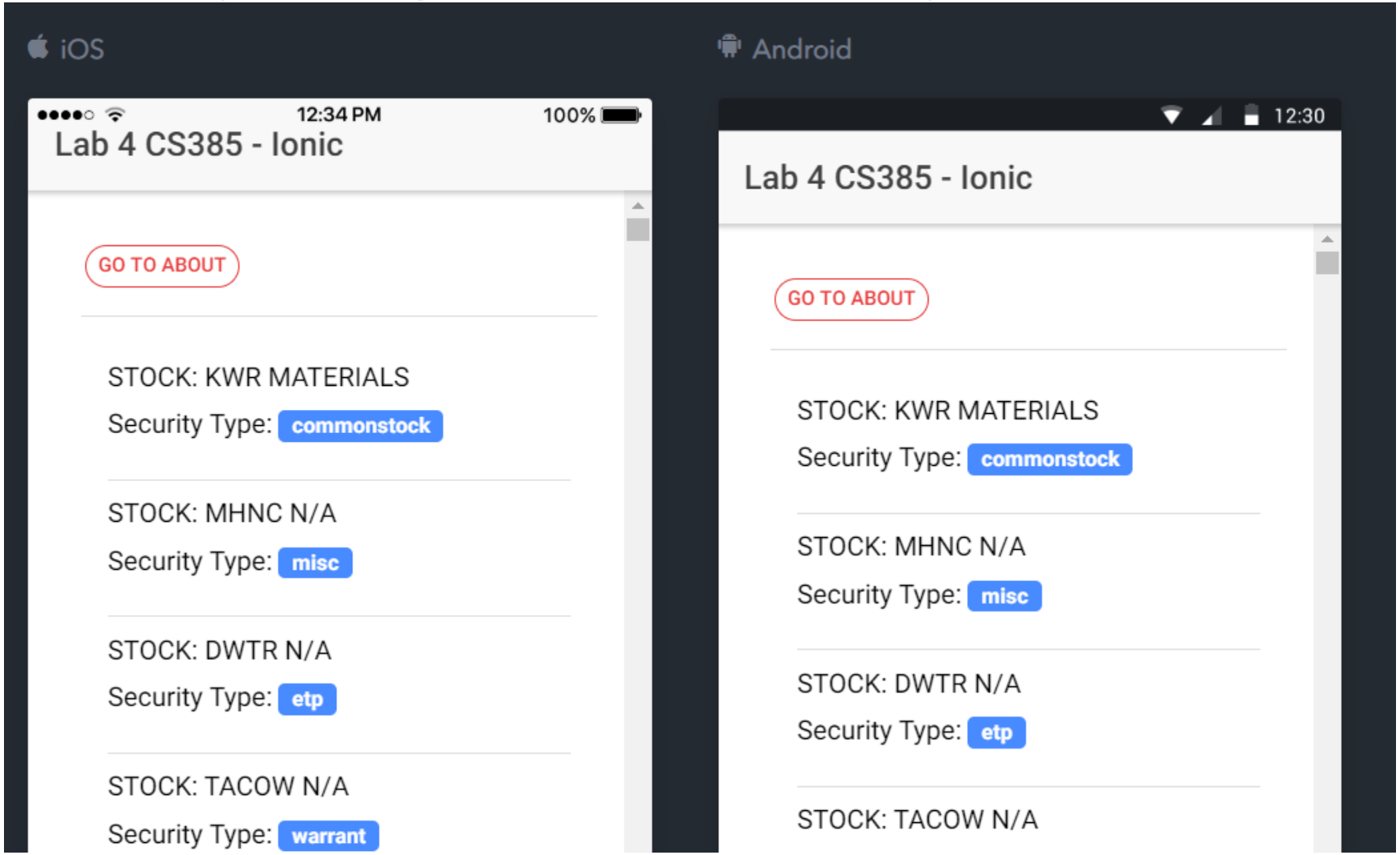
```
18
19    constructor(public navCtrl: NavController, public
20    }
21
22    ionViewDidLoad() {
23      console.log('ionViewDidLoad AboutPage');
24    }
25
26  public goToHome()
27  {
28    this.navCtrl.push(HomePage);
29  }
30
31
32
33
34  }
35
```

Step 14B – copy the Typescript function goToAbout() into the home.ts file

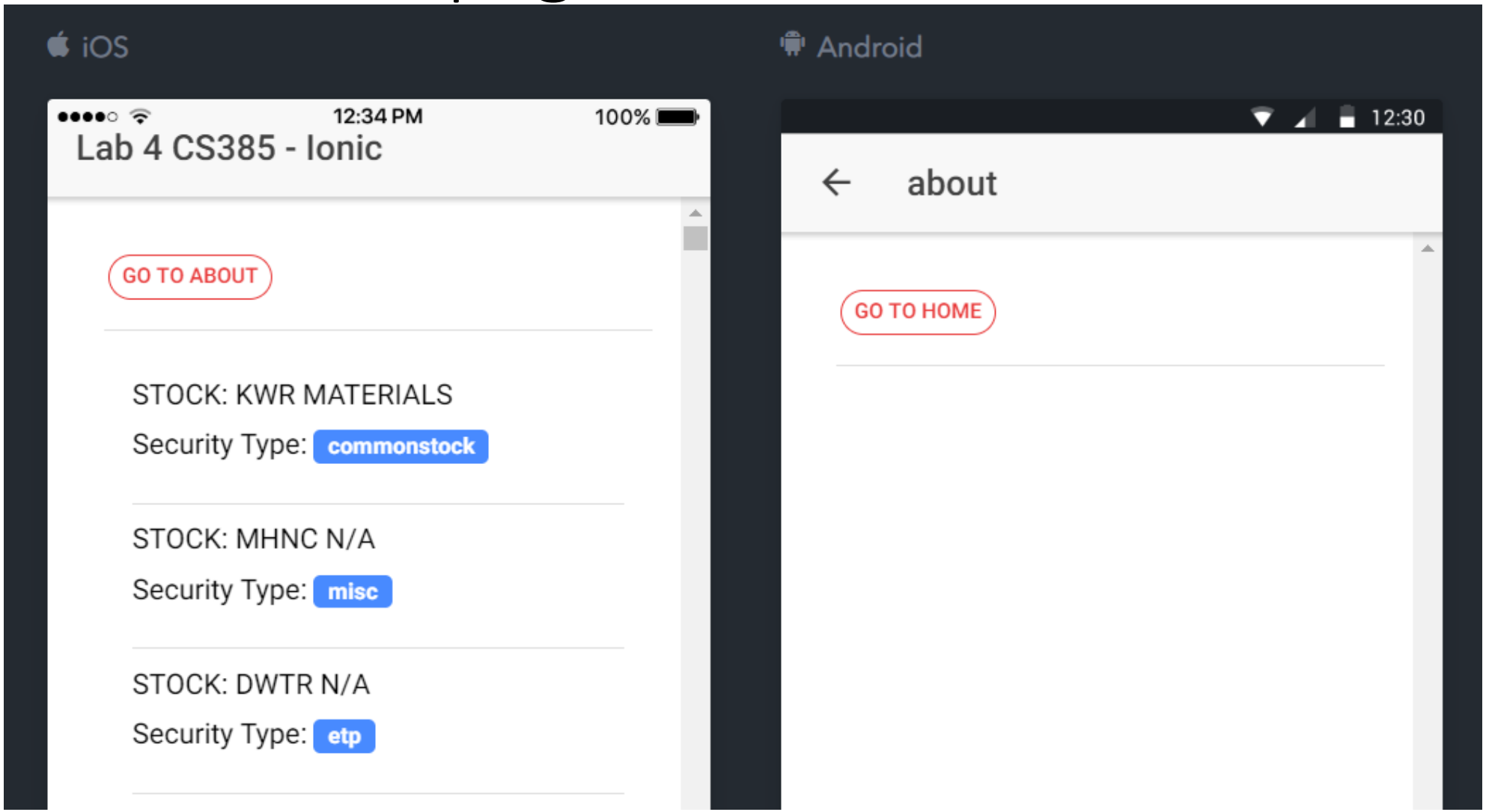
- Paste it into the file just before the last closing curly bracket.

```
47          /* assign the data from the Promise to our property
48          this.stockData = data;
49
50          console.log("Got results from the Promise");
51        });
52  }
53
54  public goToAbout()
55  {
56    this.navCtrl.push(AboutPage);
57  }
58
59
60  }
61
```

SAVE everything and refresh your browser



You should see the application MOVE to the new about.html page



Maybe put some content into about.html?

- The page is blank … but obviously we can put some text or html into this page.

- We’re done!