SEG3102 – Lab 1
Introduction to Angular

The objective of this lab is to introduce to application development with the Angular framework. The lab discusses the installation of the necessary tools for Angular and the development of a simple "*hello world*" type of application.

The source code for the lab is available in the Github repository https://github.com/stephanesome/angularIntro.git.

Installation

Node.js

Angular requires **Node.js**. Download and install the latest version from https://nodejs.org/en/.

Check that **Node.js** is properly installed by running command `node -v` in a terminal.

The **npm** package manager is also needed. It should be installed with **Node.js**.

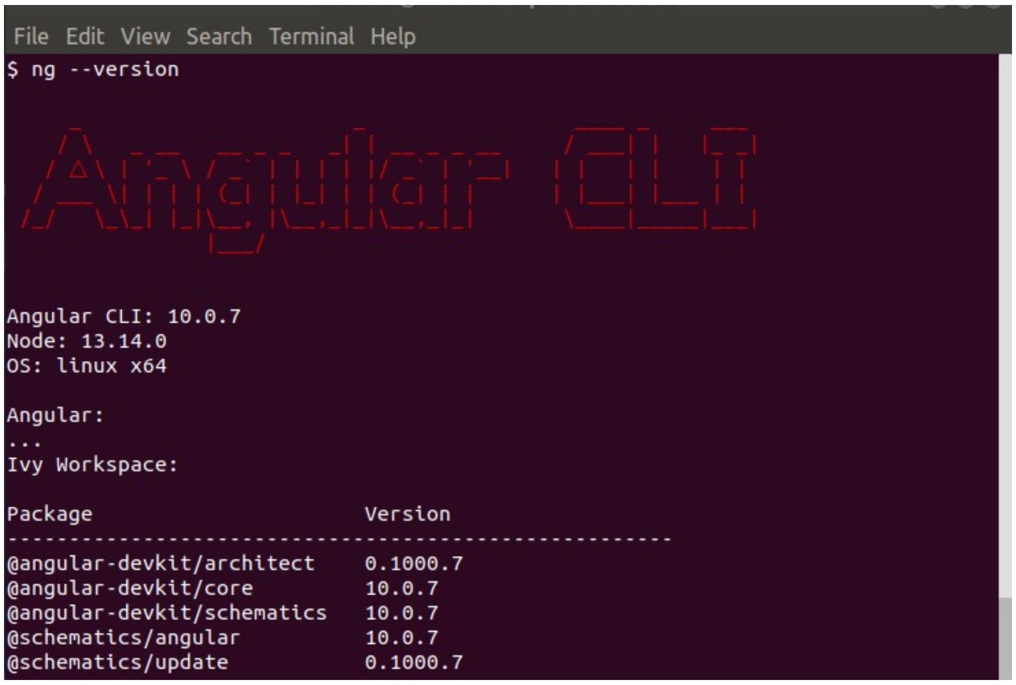
Check that **npm** is properly installed by running command `npm -v` in a terminal.

Angular CLI

The Angular CLI is used to create, test, bundle and deploy Angular projects.

Install the latest version of Angular CLI with command `npm install -g @angular/cli`.

Check the install by running command `ng -version`. You should see an output similar to the following.



```
$ ng --version

Angular CLI

Angular CLI: 10.0.7
Node: 13.14.0
OS: linux x64

Angular:
...
Ivy Workspace:

Package                      Version
------------------------------------------------------
@angular-devkit/architect    0.1000.7
@angular-devkit/core         10.0.7
@angular-devkit/schematics   10.0.7
@schematics/angular          10.0.7
@schematics/update           0.1000.7
```

Integrated Development Environment

An IDE is optional, but strongly recommended from Angular development. Several IDEs can be used including Intellij IDEA, WebStorm, Visual Studio Code, Eclipse, Atom. Make sure to set up these tools with the appropriate plugins for Angular.

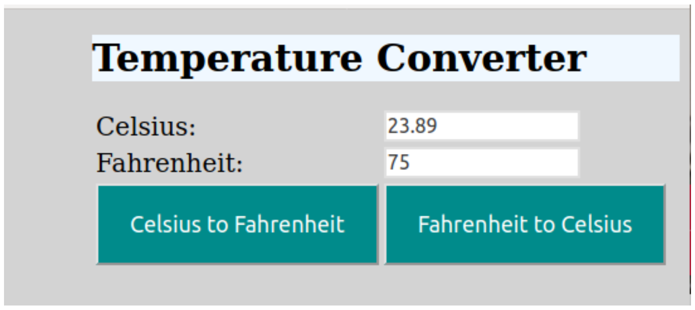
Typescript

Angular uses **Typescript**, an open-source superset of ES6 developed by Microsoft. **TypeScript** supports the features introduces in ES6 such as Classes, Modules, Fat Arrow functions (https://www.w3schools.com/js/js_es6.asp). It introduces additional features including strong typing and decorators. The following types are supported: *String*, *Number*, *Boolean*, *Array*, *Enums*, *Any*, *Void*. *Any* is a default type when typing is omitted for a variable while *Void* is used as returned type for functions with no return value.

Consult https://www.typescriptlang.org/docs for more details on Typescript.

Temperature Converter Example

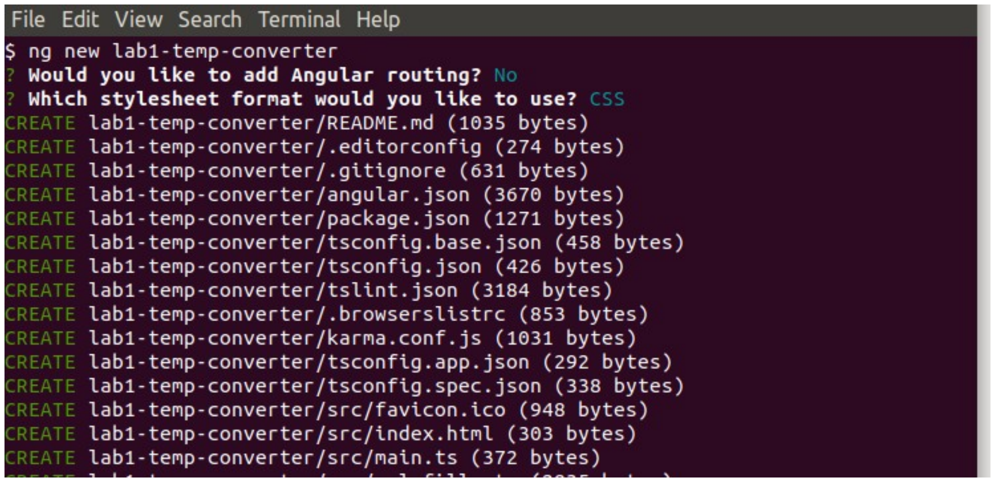
We will implement a simple Angular application for Celsius – Fahrenheit temperature conversion.



The application will allow both to convert from Celsius to Fahrenheit and from Fahrenheit to Celsius. In each case the user needs to enter a value to be converted in a text field and click on a button for the conversion. The application will then display the converted value in the other field.

Create Application

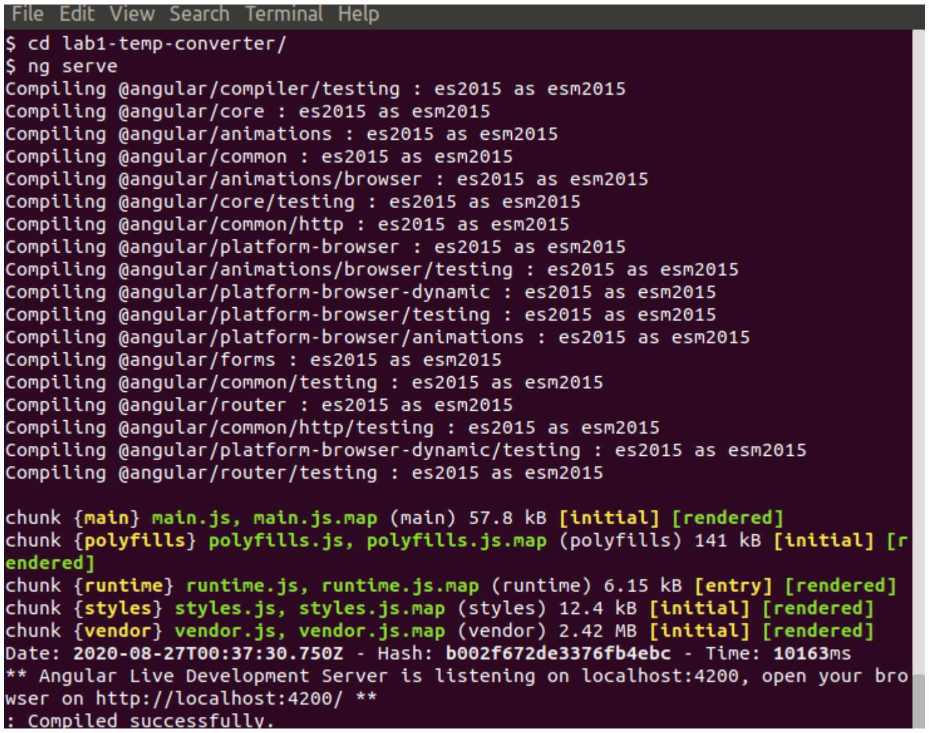
Enter command `ng new lab1-temp-converter`, respond N (no) to the addition of routing and keep CSS as stylesheet format.



```
$ ng new lab1-temp-converter
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? CSS
CREATE lab1-temp-converter/README.md (1035 bytes)
CREATE lab1-temp-converter/.editorconfig (274 bytes)
CREATE lab1-temp-converter/.gitignore (631 bytes)
CREATE lab1-temp-converter/angular.json (3670 bytes)
CREATE lab1-temp-converter/package.json (1271 bytes)
CREATE lab1-temp-converter/tsconfig.base.json (458 bytes)
CREATE lab1-temp-converter/tsconfig.json (426 bytes)
CREATE lab1-temp-converter/tslint.json (3184 bytes)
CREATE lab1-temp-converter/.browserslistrc (853 bytes)
CREATE lab1-temp-converter/karma.conf.js (1031 bytes)
CREATE lab1-temp-converter/tsconfig.app.json (292 bytes)
CREATE lab1-temp-converter/tsconfig.spec.json (338 bytes)
CREATE lab1-temp-converter/src/favicon.ico (948 bytes)
CREATE lab1-temp-converter/src/index.html (303 bytes)
CREATE lab1-temp-converter/src/main.ts (372 bytes)
```

Angular CLI will generate the necessary directory structure and setup files for the Angular application.

You may run the generated application by entering the command `ng serve` in the root directory of the project.

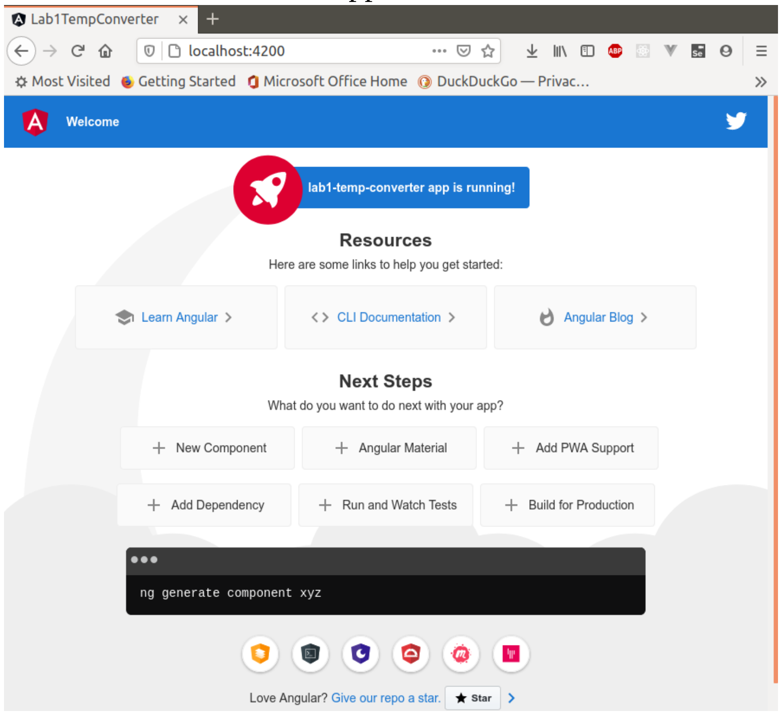


```
$ cd lab1-temp-converter/
$ ng serve
Compiling @angular/compiler/testing : es2015 as esm2015
Compiling @angular/core : es2015 as esm2015
Compiling @angular/animations : es2015 as esm2015
Compiling @angular/common : es2015 as esm2015
Compiling @angular/animations/browser : es2015 as esm2015
Compiling @angular/core/testing : es2015 as esm2015
Compiling @angular/common/http : es2015 as esm2015
Compiling @angular/platform-browser : es2015 as esm2015
Compiling @angular/animations/browser/testing : es2015 as esm2015
Compiling @angular/platform-browser-dynamic : es2015 as esm2015
Compiling @angular/platform-browser/testing : es2015 as esm2015
Compiling @angular/platform-browser/animations : es2015 as esm2015
Compiling @angular/forms : es2015 as esm2015
Compiling @angular/common/testing : es2015 as esm2015
Compiling @angular/router : es2015 as esm2015
Compiling @angular/common/http/testing : es2015 as esm2015
Compiling @angular/platform-browser-dynamic/testing : es2015 as esm2015
Compiling @angular/router/testing : es2015 as esm2015

chunk {main} main.js, main.js.map (main) 57.8 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 141 kB [initial] [r
endered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 12.4 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 2.42 MB [initial] [rendered]
Date: 2020-08-27T00:37:30.750Z - Hash: b002f672de3376fb4ebc - Time: 10163ms
** Angular Live Development Server is listening on localhost:4200, open your bro
wser on http://localhost:4200/ **
: Compiled successfully.
```

Angular CLI integrates a Development Server and deploys the application at the URL http://localhost:4200.

Navigate to the URL http://localhost:4200 to view the application.



The generated application provides links to tutorials and documentation on Angular.

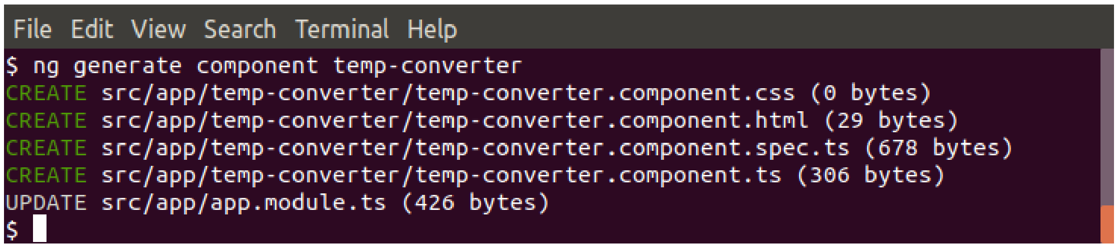
Open Application in IDE

I'll be using Intellij IDEA to develop the application. Although we could develop our applications using a simple word processor, using an IDE with features such as code completion or error checking makes the job more productive.

- **File -> Open** and browse to select the project root directory.

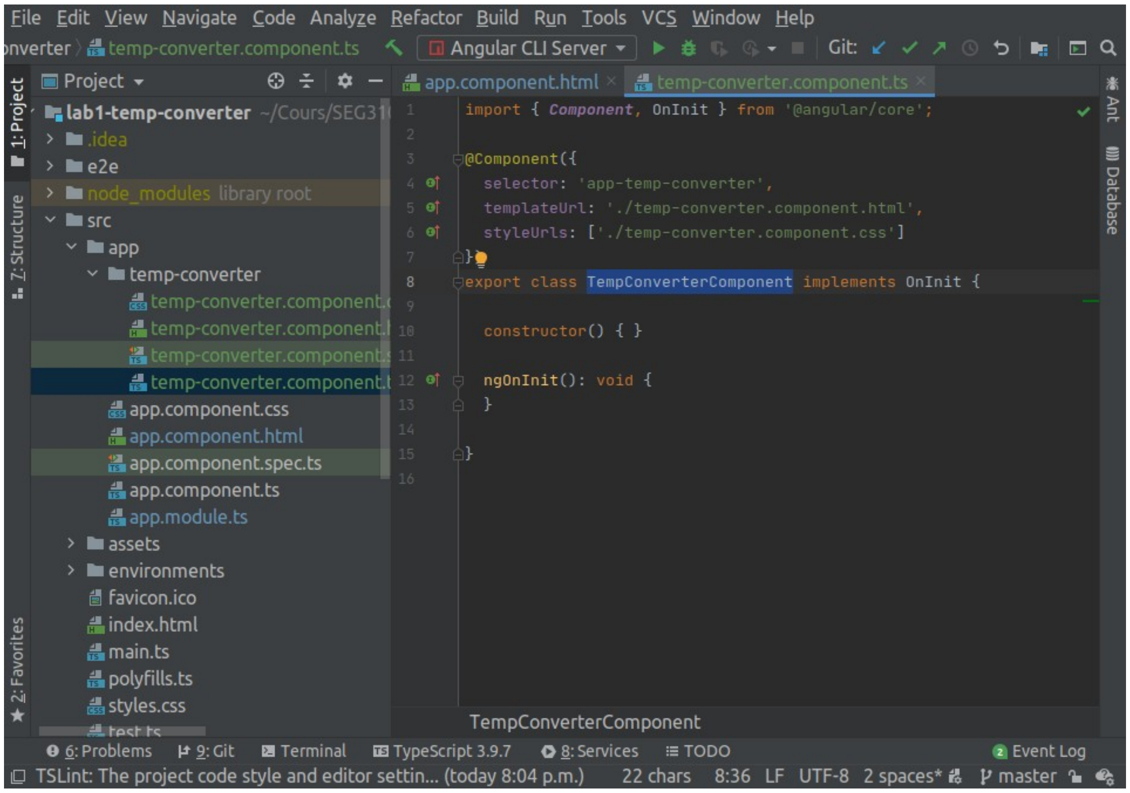
Create Temperature Converter component

- Create a component with command `ng generate component temp-converter`.



```
$ ng generate component temp-converter
CREATE src/app/temp-converter/temp-converter.component.css (0 bytes)
CREATE src/app/temp-converter/temp-converter.component.html (29 bytes)
CREATE src/app/temp-converter/temp-converter.component.spec.ts (678 bytes)
CREATE src/app/temp-converter/temp-converter.component.ts (306 bytes)
UPDATE src/app/app.module.ts (426 bytes)
$
```

The command generates component `TempConverter` that we will use for the temperature conversion.


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
nverter › temp-converter.component.ts
Angular CLI Server
Git:
Project
lab1-temp-converter ~/Cours/SEG31
.idea
e2e
node_modules library root
src
app
temp-converter
temp-converter.component.
temp-converter.component.
temp-converter.component.
temp-converter.component.
app.component.css
app.component.html
app.component.spec.ts
app.component.ts
app.module.ts
assets
environments
favicon.ico
index.html
main.ts
polyfills.ts
styles.css
test.ts
app.component.html
temp-converter.component.ts
1: Project
7: Structure
2: Favorites
Ant
Database
TempConverterComponent
6: Problems
9: Git
Terminal
TypeScript 3.9.7
8: Services
TODO
Event Log
TSLint: The project code style and editor settin... (today 8:04 p.m.)
22 chars 8:36 LF UTF-8 2 spaces* master

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-temp-converter',
  templateUrl: './temp-converter.component.html',
  styleUrls: ['./temp-converter.component.css']
})
export class TempConverterComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

Setup Temperature Converter component

- Setup HTML template of the `TempConverter` Component. Edit src/app/temp-converter/temp-converter.component.html so that it is as follow.

```
1.  <table>
2.    <tr>
3.      <td>
4.        <label for="celsius">Celsius:</label>
5.      </td>
6.      <td>
7.        <input name="celsius" id="celsius" value="{{celsiusValue | number: '1.0-2'}}" #celsius>
8.      </td>
9.    </tr>
10.   <tr>
11.     <td>
12.       <label for="fahrenheit">Fahrenheit:</label>
13.     </td>
14.     <td>
15.       <input name="fahrenheit" id="fahrenheit" value="{{fahrenheitValue | number: '1.0-2'}}" #fahrenheit>
16.     </td>
17.   </tr>
18.   <tr>
19.     <td>
20.       <button (click)="convertCelsius(celsius.value)">Celsius to Fahrenheit</button>
21.     </td>
22.     <td>
23.       <button (click)="convertFahrenheit(fahrenheit.value)">Fahrenheit to Celsius</button>
24.     </td>
25.   </tr>
26. </table>
```

The values of the input elements at line 7 and 15 are bound to properties in the component class using string interpolation. We use the number pipe to display the values with no more than 2 decimals.

The input elements are associated with template reference variables #celsius and #fahrenheit respectively. These template reference variables are used to pass the input values to the button click event handlers in lines 20 and 23.

- Setup class `TempConverterComponent`. Edit src/app/temp-converter/temp-converter.component.ts so that it is as follow.

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-temp-converter',
  templateUrl: './temp-converter.component.html',
  styleUrls: ['./temp-converter.component.css']
7.   })
8.   export class TempConverterComponent {
9.     celsiusValue = 0;
10.    fahrenheitValue = 0;
11.    constructor() { }
12.
13.    convertCelsius(value: string): void {
14.      this.celsiusValue = Number(value);
15.      this.fahrenheitValue = ((this.celsiusValue * 9) / 5) + 32;
16.    }
17.
18.    convertFahrenheit(value: string): void {
19.      this.fahrenheitValue = Number(value);
20.      this.celsiusValue = ((this.fahrenheitValue - 32) * 5) / 9;
21.    }
22. }
```

The decorator specifies meta-data for the component. The selector property in line 4 defines the tag for the component, to be use in HTML templates to include the component.

Properties templateUrl in line 5 and styleUrls in line 6 specifies files for the component HTML template and style sheets.

Class `TempConverterComponent` defines two properties celsiusValue and fahrenheitValue that passes values to the HTML template using string interpolation.

Functions convertCelsius at lines 13-16 and convertFahrenheit at lines 18-21 are the button click event handlers. These functions perform the required conversion and set the celsiusValue and fahrenheitValue properties.

- Add styling.

Edit src/styles.css so that it is as follow.

```
body {
  background-color: lightgray;
  margin-left: 70px;
  margin-top: 20px;
}
```

src/styles.css specifies styles that apply to the whole application.

Edit src/app/app.component.css as follow for styling of the Root component.

```
h1 {
  font-size: large;
  font-family: serif;
  font-weight: bold;
}
```

And edit src/app/temp-converter/temp-converter.component.css as follow for styling of the `TemperatureConverter` component.

```
input {
  border-style: double;
  width: 150px;
}
button {
  background-color: darkcyan;
  padding: 20px 25px;
  font-size: 18px;
  color: white;
}

label {
  font-size: 20px;
}
```

Tests

Angular CLI generates components with associated tests.

`src/app/temp-converter/temp-converter.component.spec.ts` contains the necessary setup to unit test the Temperature Converter component with the Jasmine test framework. Add the following test cases.

```
it('should convert 0 celsius to 32 fahrenheit', () => {
  const tempval = '0';
  component.convertCelsius(tempval);
  expect(component.fahrenheitValue).toBeCloseTo(32);
});

it('should convert -100 fahrenheit to -73.33 celsius', () => {
  const tempval = '-100';
  component.convertFahrenheit(tempval);
  expect(component.celsiusValue).toBeCloseTo(-73.33);
});
```

Setup Root Component

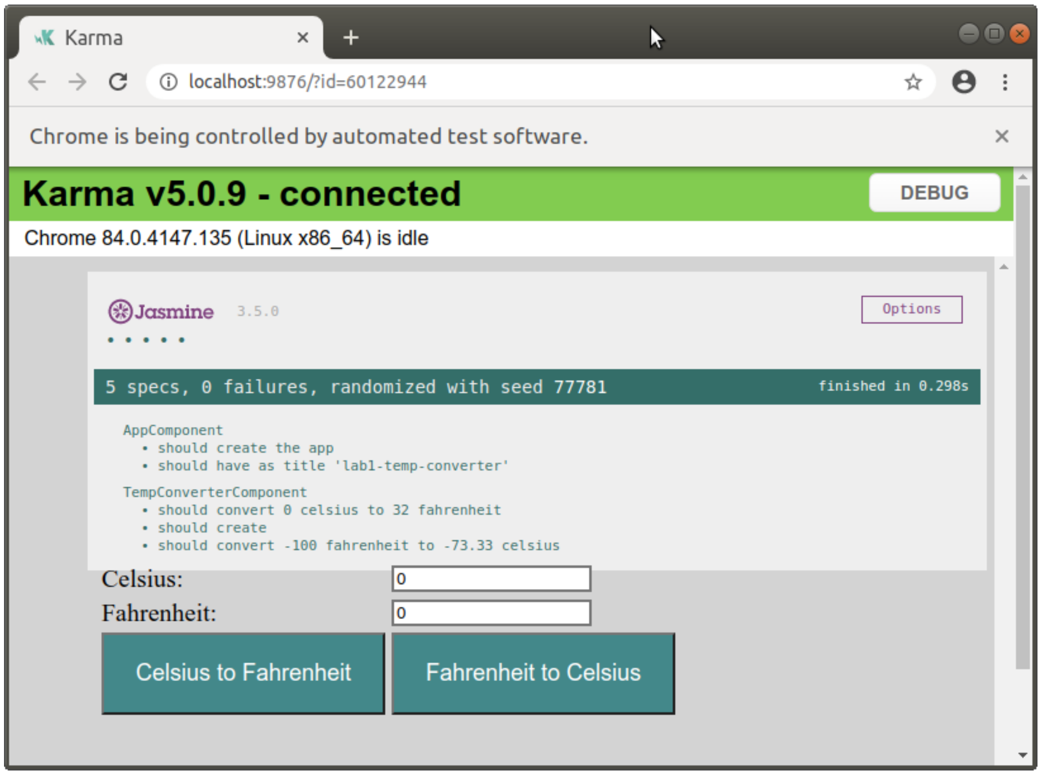
- Remove the generated content in the root component AppComponent HTML template (`/src/app/app.component.html`) and replace with the following.

```
<div>
  <h1>Temperature Converter</h1>
  <app-temp-converter></app-temp-converter>
</div>
```

The HTML template for the Root component includes the selector tag for the TemperatureConverter component. This will instantiate and render the component where the selectors are specified.

Running Tests

We run all the tests with command `ng test`. This include Jasmine unit tests as well as end-to-end browser driven tests.



Chrome is being controlled by automated test software.

Karma v5.0.9 - connected

Chrome 84.0.4147.135 (Linux x86_64) is idle

```
5 specs, 0 failures, randomized with seed 77781                finished in 0.298s

AppComponent
  • should create the app
  • should have as title 'lab1-temp-converter'
TempConverterComponent
  • should convert 0 celsius to 32 fahrenheit
  • should create
  • should convert -100 fahrenheit to -73.33 celsius
```

Celsius: 0

Fahrenheit: 0

Celsius to Fahrenheit | Fahrenheit to Celsius

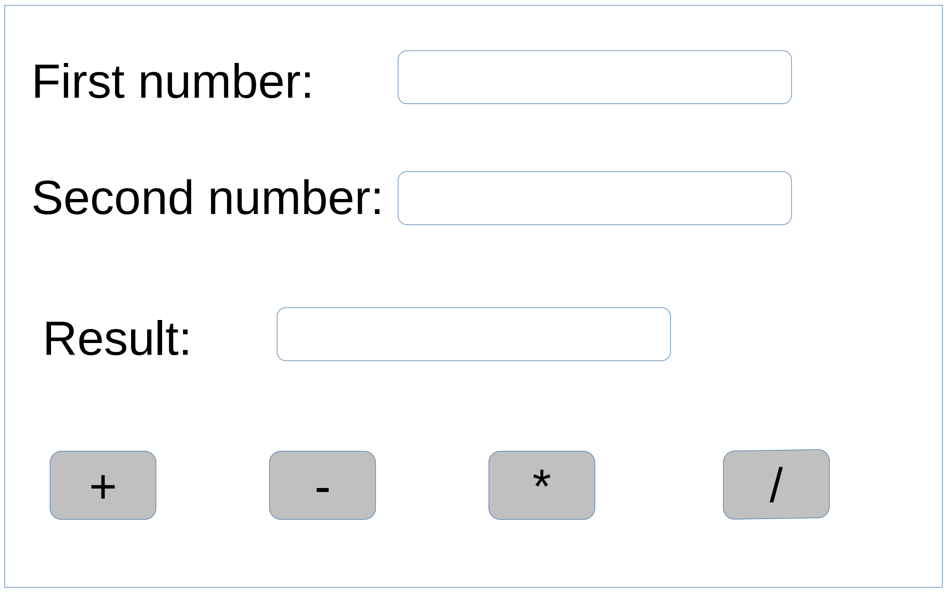
Running the Application

Enter command `ng serve` to build and start the application. Then navigate to the URL http://localhost:4200.

Exercise

The following exercise is the deliverable for the lab. Complete and check-in the code to Github Classroom before the deadline. Only this exercise will be evaluated.

Implement an Angular application for a basic Calculator. The application shall show an interface similar to the follow



The user will then be able to enter two number and select an operation. Upon submission, the application shall perform the required operation and show the result.