Angular
Lab 5
Pipes

1 Lab setup

Make sure you have the following items installed

- Latest version of Node and NPM (note: labs are tested with Node v16.13 and NPM v8.1 but should work with later versions with no or minimal changes)
- Latest version of TypeScript (note: labs are tested with TypeScript v4.5.4 but should work with later versions with no or minimal changes)
- Latest version of Angular (note: labs are tested with Angular v13.1 but should work with later versions with no or minimal changes)
- Visual Studio Code (or a suitable alternative IDE for Angular/TypeScript)
- A suitable text editor (Notepad ++)
- A utility to extract zip files (7-zip)

In each of the main lab folders, there are two subfolders: `changes` and `final`.

- The `changes` subfolder holds the source code and other related files for the lab. The creation of the Angular app will proceed in a step-wise fashion in order to clearly demonstrate the various features of the framework. The various project source files in this folder are numbered in a way to reflect this gradual construction. For e.g. `xxx.v1.html`, `xxx.v2.html`, etc represents successive changes that are to be made to the `xxxx.html`

- The `final` subfolder holds the complete Angular project. If there was some issue in completing the step-wise construction of the app using the files from the `changes` subfolder, you can use this to run the final working version.

The project folder in `final` is however missing the crucial `node_modules` subfolder which contains all the dependencies (JavaScript libraries) that your Angular app needs in order to be built properly. These dependencies are specified in the `package.json` file in the root project folder. They are omitted in the project commit to GitHub because the number and size of the files are prohibitively large.

In order to run the app via the Angular development server (`ng serve`), you will need to create this subfolder containing the required dependencies. There are two ways to accomplish this:

a) Copy the project root folder to a suitable location on your local drive and run `npm install` in a shell prompt in the root project folder. NPM will then create the `node_modules` subfolder and download the required dependencies as specified in `package.json` to populate it. This process can take quite a while to complete (depending on your broadband connection speed)

b) Alternatively, you can reuse the dependencies of an existing complete project as most of the projects for our workshop use the same dependencies. In this case, copy the `node_modules` subfolder from any existing project, and you should be able to immediately start the app. If you encounter any errors, this probably means that there are some missing dependencies or Angular version mismatch, in which case you will then have to resort to `npm install`.

The lab instructions here assume you are using the Chrome browser. If you are using a different browser (Edge, Firefox), do a Google search to find the equivalent functionality.

2 Generating a new Angular project

In either a separate command prompt (or shell terminal) or in an embedded terminal in Visual Studio Code, create a new Angular project with:

```
ng new pipesdemo
```

Press enter to accept the default values for all the question prompts that follow regarding routing and stylesheet.

Navigate into the root project folder from the terminal with:

```
cd pipesdemo
```

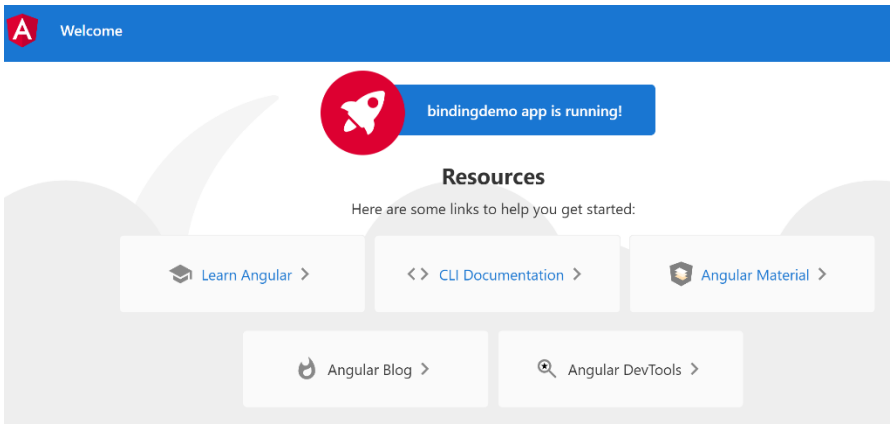
Build the app and serve it via Angular's live development server by typing:

```
ng serve
```

The final output line from this should indicate that the compilation process was successful and identify the port that the live development server is currently serving up the Angular app dynamically from (by default this will be port 4200)

Open a browser tab at:
http://localhost:4200/

You should be able to see the default landing page for all new autogenerated Angular projects.



To stop the development server, type Ctrl+C in the terminal window that it is running in. You can restart again anytime by typing `ng serve` in the project root folder.

3 Working with the date pipe

The source code for this lab can be found in the `changes` subfolder of `Pipes-Demo`

Pipes are simple functions used in template expressions to accept an input value and return a transformed value. They are used to transform strings, currency amounts, dates, and other data to a suitable format for display. Pipe functions are displayed to the right of the input values they are transforming and are separate by the | pipe operator.

Modify the following files in `src/app` with the latest update from `changes`:

```
app.component-v1.html
app.component-v1.ts
```

The date pipe accepts three optional arguments that are specified after it using :

```
{{ date_expression | date [ : format [ : timezone [ : locale ] ] ] }}
```

If no predefined format is specified, the default is `mediumDate`.

For more info on creating dates in JavaScript:
https://css-tricks.com/everything-you-need-to-know-about-date-in-javascript/
For specifying dates as Strings, use the format shown here:
https://www.w3.org/TR/NOTE-datetime

Modify the following files in `src/app` with the latest update from `changes`:

```
app.component-v1-2.html
```

Here we are demonstrating the use of the different date/time and date only formats
https://angular.io/api/common/DatePipe#pre-defined-format-options

as well as the ability to customize the display if none of the predefined format options meets our needs:
https://angular.io/api/common/DatePipe#custom-format-options

Modify the following files in `src/app` with the latest update from `changes`:

`app.component-v1-3.html`

To find the time zone set on the local development machine:
https://www.nextofwindows.com/windows-10-tip-where-to-check-and-set-time-zone-on-my-computer

The time zone can be specified as the UTC +/- x as the 2nd argument to the pipe function.
You can check on the time zones for global locations here:
https://www.timeanddate.com/worldclock/converter.html

References:

https://www.tektutorialshub.com/angular/formatting-dates-with-angular-date-pipe/

https://angular.io/api/common/DatePipe

If no predefined format is specified, the default is `mediumDate`.

Timezone abbreviations (2nd argument to the Date pipe): https://www.timeanddate.com/time/zones/

To change the LOCALE_ID for locale (3rd argument to the Date pipe):
https://angular.io/api/core/LOCALE_ID

https://angular.io/guide/component-overview#creating-a-component

4 Working with the decimal pipe

Modify the following files in `src/app` with the latest update from `changes`:

`app.component-v2.ts`

`app.component-v2.html`

The date pipe accepts 2 optional arguments that are specified after it using :

```
{{ value_expression | number [ : digitsInfo [ : locale ] ] }}
```

The digitsInfo parameter specifies the number of integer digits to display before and after the decimal point:
https://angular.io/api/common/DecimalPipe#digitsinfo

References:

https://angular.io/api/common/DecimalPipe

https://www.angularjswiki.com/angular/angular-decimal-pipe/

5 Working with the currency pipe

Modify the following files in `src/app` with the latest update from `changes`:

`app.component-v3.ts`

`app.component-v3.html`

The current pipe accepts four optional arguments that are specified after it using :

```
{{ value_expression | currency [ : currencyCode [ : display [ :
digitsInfo [ : locale ] ] ] ] }}
```

The list of currency codes and symbols for the major world currencies:
https://www.xe.com/symbols.php

Modify the following files in `src/app` with the latest update from `changes`:

`app.component-v3-2.ts`

`app.component-v3-2.html`

Each locale has a specific format for displaying a number, and this will also vary depending on the specific currency being displayed:
https://phrase.com/blog/posts/number-localization/#Separators_Breaking_Up_Large_Numbers

The list of major global locales are listed here:
https://www.science.co.il/language/Locale-codes.php

To find the current system locale:
https://winaero.com/find-current-system-locale-in-windows-10/

References:

https://angular.io/api/common/CurrencyPipe

https://www.angularjswiki.com/angular/angular-currency-pipe-formatting-currency-in-angular/

6 Working with the json pipe

Modify the following files in `src/app` with the latest update from `changes`:

`app.component-v4.html`

`app.component-v4.ts`

There are a few other built-in pipes that can be used:
https://angular.io/api/common#pipes

References:

https://angular.io/api/common/JsonPipe

https://www.concretepage.com/angular-2/angular-2-json-pipe-example#jsonpipe

7 Creating custom pipes

Modify / add the following files in `src/app` with the latest update from `changes`:

`app.component-v5.html`

`app.component-v5.ts`

`app.module-v5.ts`

`currency-converter.ts`

We can create our own custom pipes to implement customized logic specific to our application to transform values that are to be displayed in the template.
The steps involved are:

a) Create a class for the pipe function and mark it with the @pipe decorator.
b) Give a name for the pipe in the name meta data of the @pipe decorator.
c) The pipe class must implement the PipeTransform interface.
d) Implement the transform method of this interface. The first parameter to the transform method is the value before the pipe operator that is to be passed into the pipe function. The implementation of the transform method must transform the value and return the result. You can add any number of additional arguments to the transform method as required.
e) Declare the pipe class in the root module
f) Use the custom pipe in the template just like the standard built-in pipes

Here we implement a simple custom pipe to perform currency conversion between MYR and USD

We can also chain pipes through repeated use of the pipe operator so that the output from one pipe becomes the input to another pipe.

References:

https://www.tektutorialshub.com/angular/angular-custom-pipes/

https://angular.io/guide/pipes#creating-pipes-for-custom-data-transformations