Angular Framework

Angular Framework is used to develop front end applications, it allows you to create applications which are called as client applications for variety of platforms like Mobiles, Desktops, Web and so on.

It is created by Google.

What is framework

Framework helps to build complex application easily by providing all the common features for an application inbuilt so that developers don’t have to write code for these common features, some of the common features most of the application needs are:

* Type conversion
* Internationalization
* Exception Handling

Angular Framework is a framework for front-end it uses two languages to develop the front end applications & it is used to develop single page application the two languages are:

1. HTML for displaying
2. Typescript for writing the logics

Typescript is a superset of JavaScript which is better than the Java Script

Note: Angular Framework uses CSS for styling

Single Page Application:

Single Page Applications will have one single page where you can perform all the tasks, it loads only part of the page when any action is performed and doesn’t refresh the entire page, ex: Gmail, Angular official website, React.js official website, Git Hub.

Single Page Application is much faster compare to multiple page applications, because in single page application you need to update only small part of the page not the entire page.

Software required

1. Node.js
2. Visual Studio Code
3. Angular CLI (Angular Command Line Interface) toolkit

Angular CLI

It is a toolkit released by Google to improve the development process

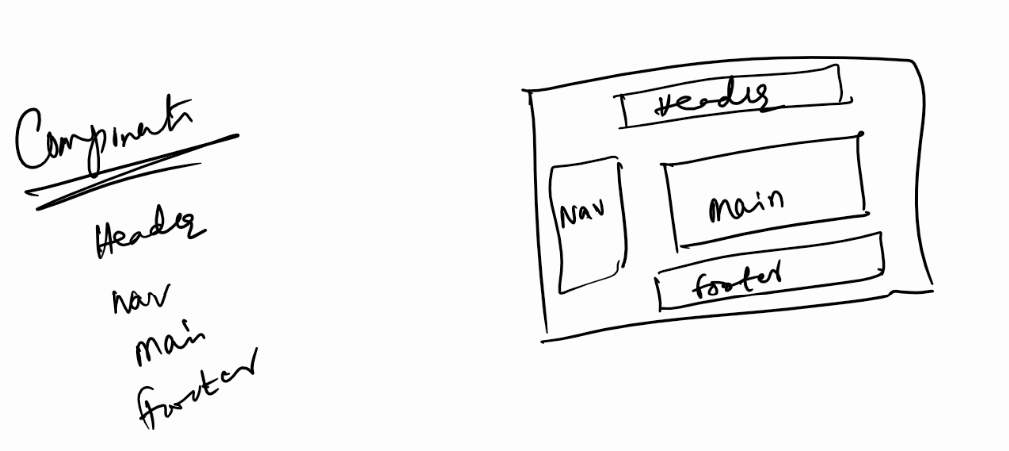
* It allows you to create a ready to run and build angular project
* It will have an inbuilt typescript compiler in the project which automatically compiles the typescript code to javascript code
* It will provide a live reload feature which will refresh the browser while you are typing the code
* It will provide an embedded server to run your angular application
* It will provide lot of angular commands to accelerate the development process, so that developers work will be reduced

How angular can create a single page application

Angular uses component to create the UI, these components are independent and reusable with other components.

Components: These are UI’s which you see in the web page, they can be created independently and nest in other components.

A single page application is a collection of multiple components



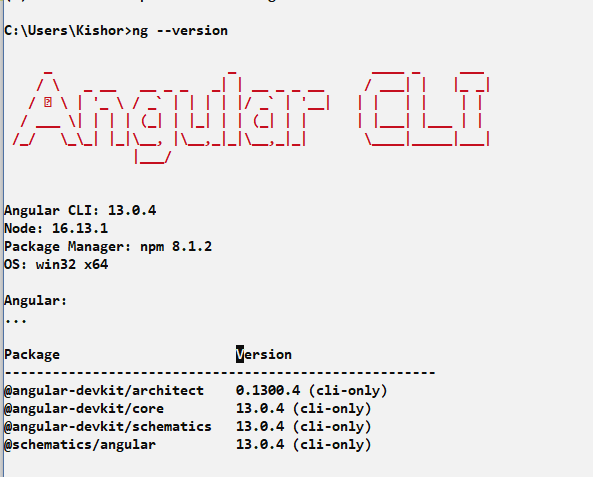
Angular CLI Installation

We should have node & npm installed already, because you will install angular cli using npm.

>> npm install -g @angular/cli

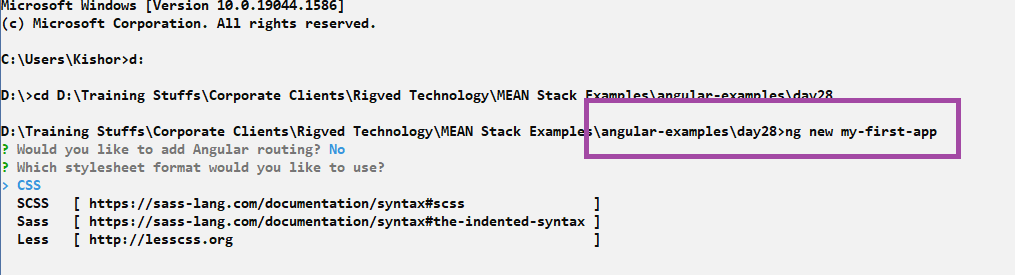
Note: Angular CLI is a one time installation, it gives you a command called ‘ng’ using which you can create angular applications & also run angular applications

After installing you can use ng --version

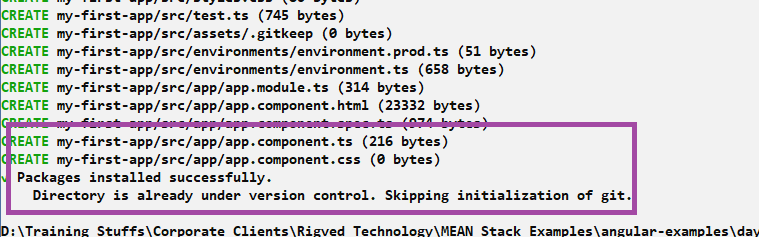


Command used to create angular project

>> ng new project-name



Once you install you will get the success confirmation



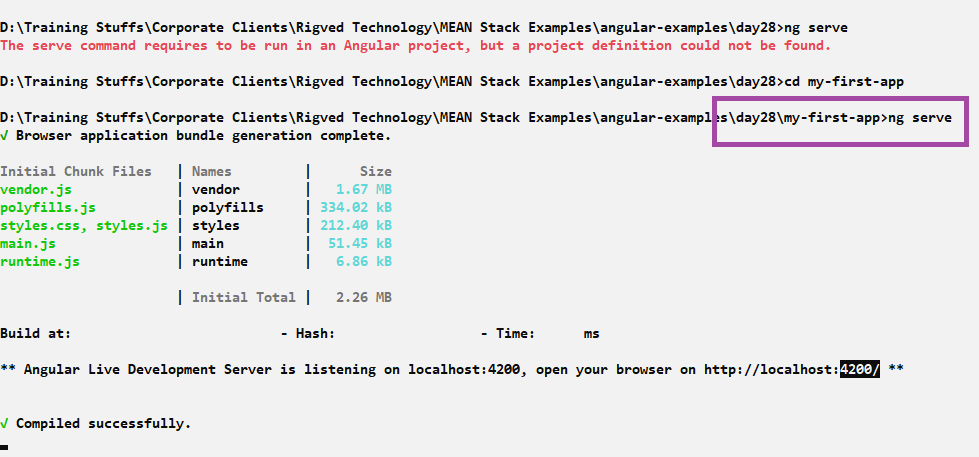
Purpose of *ng new <<app-name>>*

ng is an angular command

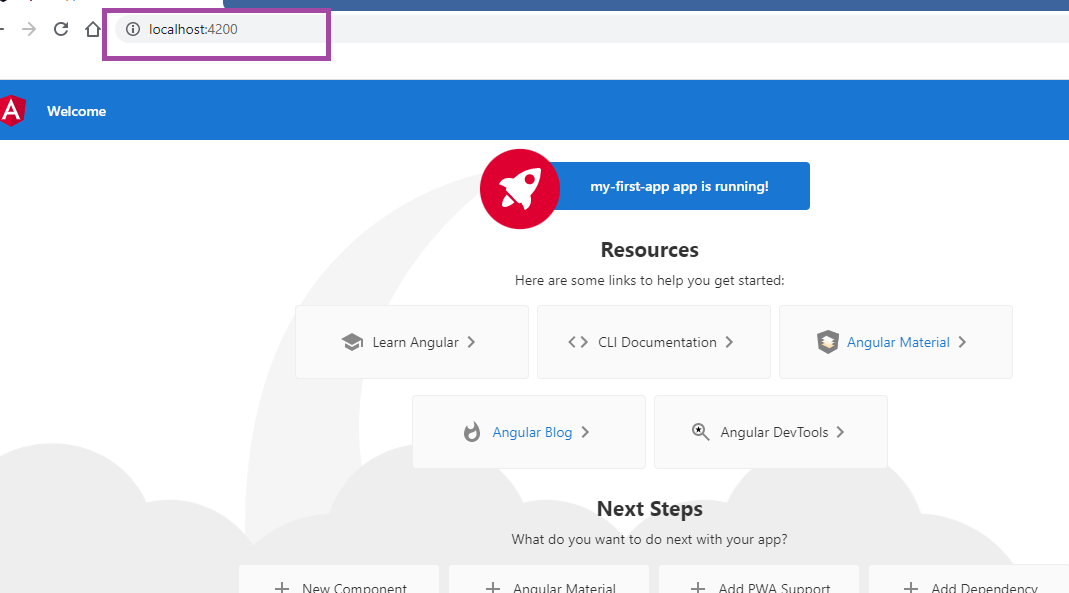
new is a keyword to create a new project

<<app-name>> must be the project name

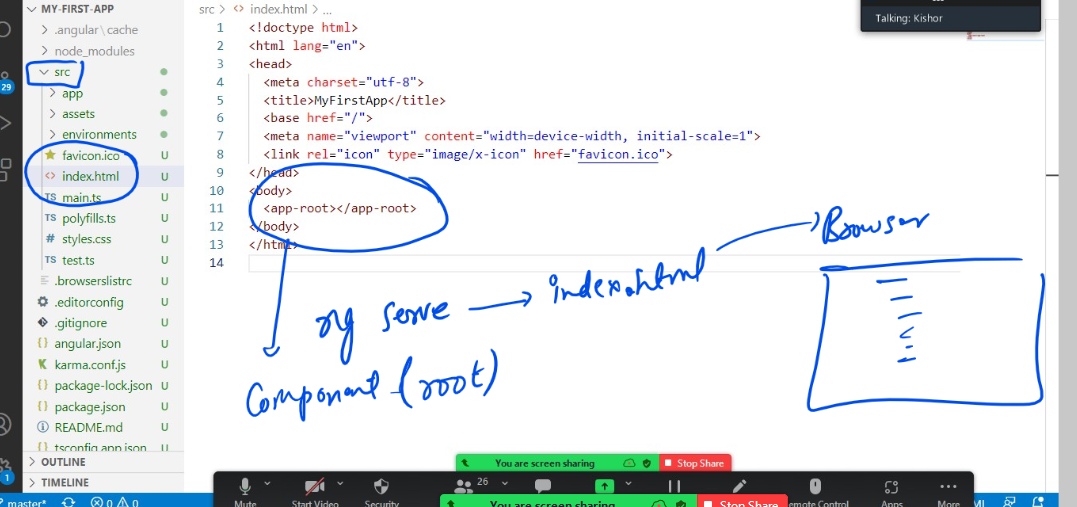
Navigate to the project & use ng serve to run your project



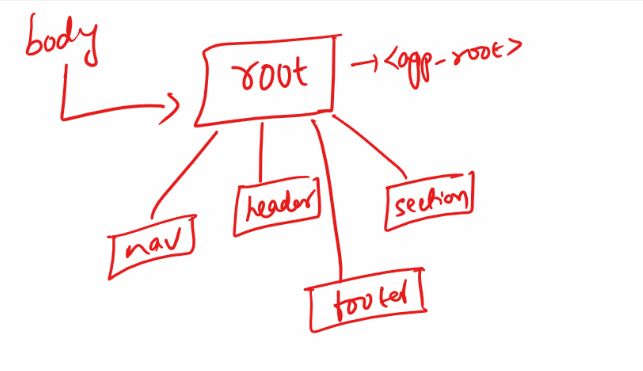
By default the project runs in 4200, which you can enter in the browser and by default every new project will have some default content which can be changed by adding UI’s in angular application according to your requirement.



You can open angular project folder from the VS Code



The moment you enter ng serve, angular loads index.html in the browser, that is only file which is loaded in the browser, but everything happens inside the index.html <app-root>, which is a root component, this root component can nest any number of components



Every angular application must have one root component and it be only one root component all the other components are part of this root component.

<app-root>: This is the tag name for root component, when you use this tag the content of root component is displayed

Where’s the content of this <app-root>

src/app/app.component.html: This file has the content for <app-root>

How does angular knows <app-root> content is in app.component.html

It is mentioned in the src/app/app.component.ts

Two important files to know

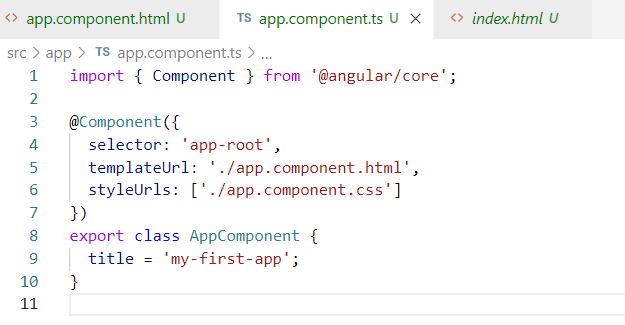
app.component.html: It has root component content

app.component.ts: It has @Component({}) decorator which specifies the tag name for the root component and content for the tag

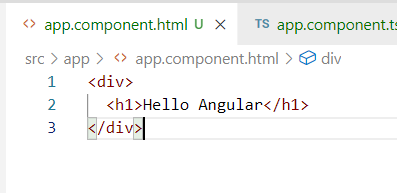
i.e., @Component({selector:’app-root’, templateUrl:’./app.component.html’})

@Component: It is a decorator which must written on top of the class when you want the class to behave like a UI component

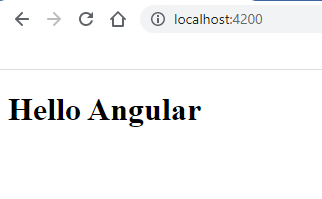
Currently src/app/app.component.ts has below code not changed



Modify the src/app/app.component.html to have a simple content



Angular can detect the changes and reload browser automatically



Note: The only file loaded in the browser is index.html & it must never be modified

Activity:

1. Try the above example as it is
2. Create a new project with the name my-second-app inside day28 folder and remove everything of src/app.component.html and add {{title}}.

Angular building blocks

1. Components
2. Modules
3. Databinding
4. Pipes
5. Services
6. Routers

Components: These are visible part in the page

Modules: This represents functionality of an entire application or part of the application, an angular application is a collection of all the building blocks which are part of an angular module called root module

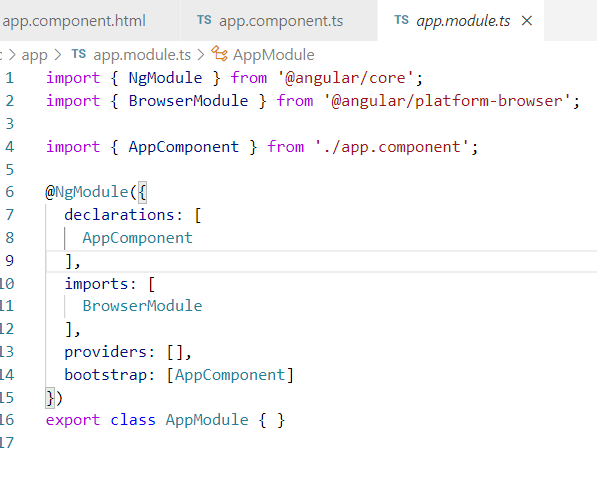
Root module will have the entries of all the components, pipes, routers, services an so on. It represents an entire application.

Your angular project also have a root module in src/app/app.module.ts

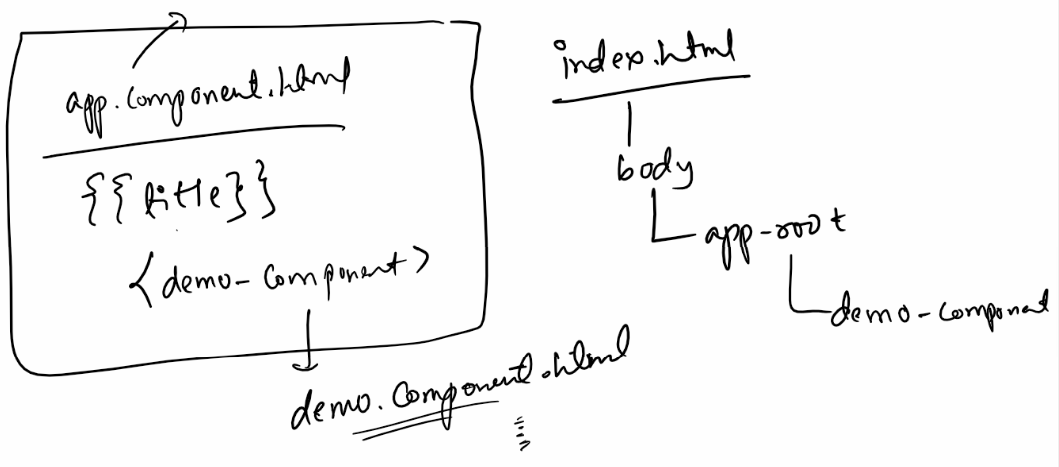
This module is launched when you enter ng serve, this takes care of loading all the elements of the application.

@NgModule({}): It is decorator used to create the module

By default app.module.ts will have below entries



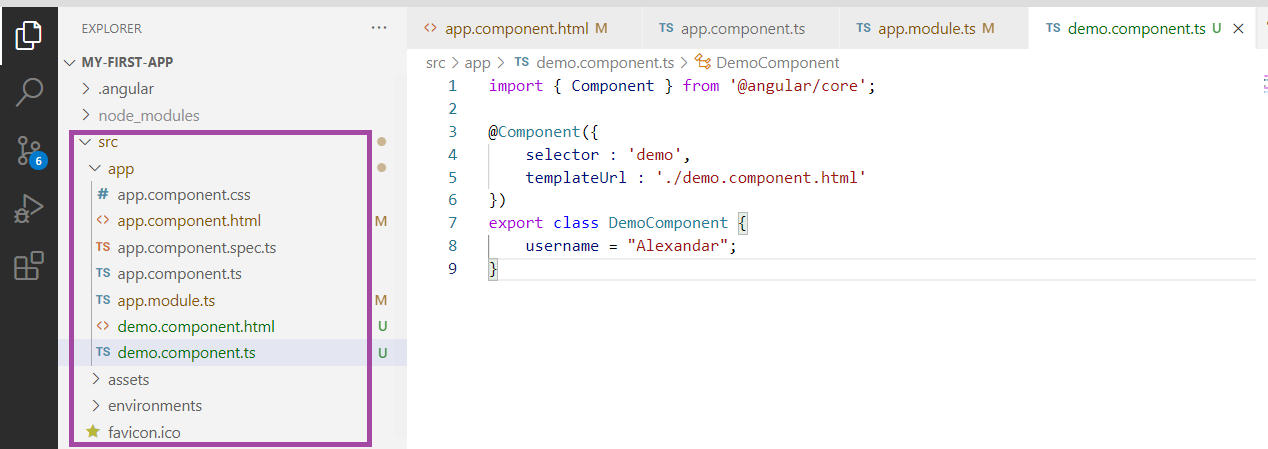
Here the declarations property mentions all the components you create in your application, any components you create is recognized by angular only if you declare in AppModule.



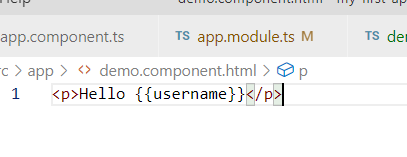
Creating our own components

1. Create a ts file
2. Create an html file
3. Mention the component in the app.module
4. Add your component(<demo>) in another component(app.componen.html)

app/demo.component.ts



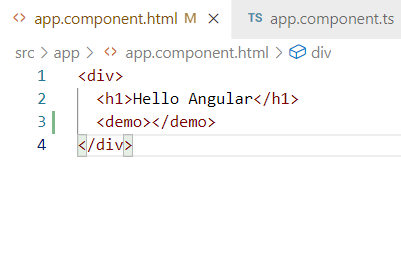
app/demo.component.html



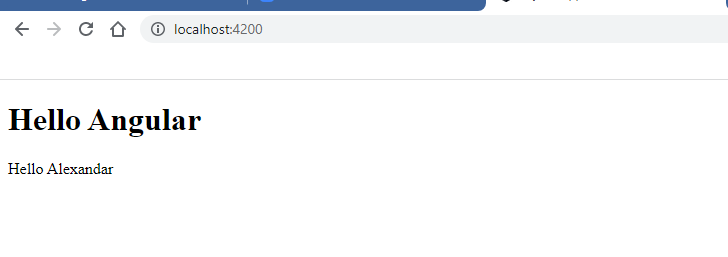
app/app.module.ts



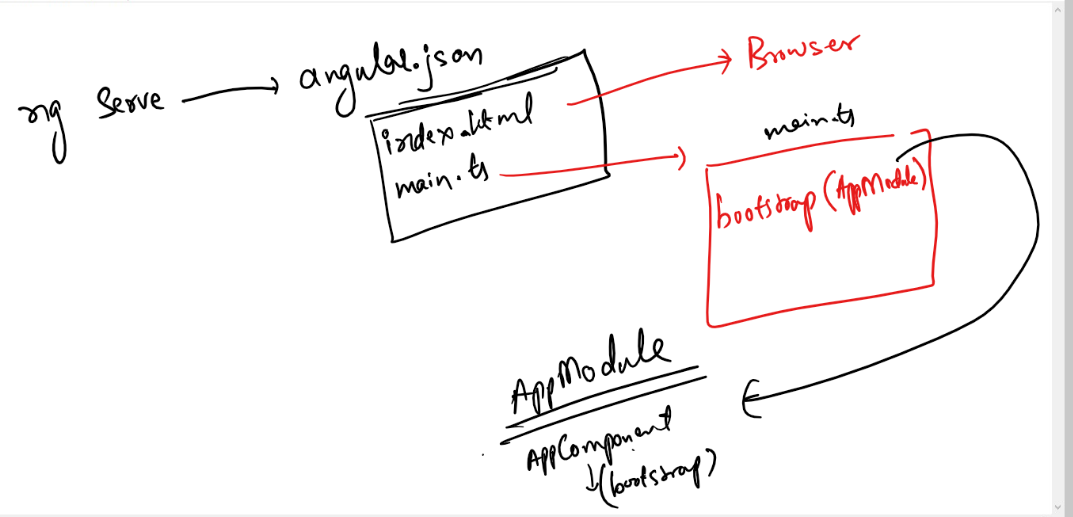
app/app.component.html



Output:



What happens when you use ng serve



The ng serve is a script that looks for angular.json file which is a configuration file used by angular there it finds index.html and main.ts file as an entry point file, the index.html will be loaded in the browser & main.ts will be executed, the main.ts will have a code to bootstrap AppModule, since the AppModule represents everything of your application the main.ts is the one who loads your entire application by loading AppModule, the AppModule will have @NgModule({}) decorator that declares all the components, other building blocks of angular but bootstraps the root components as it is the very first component appears in the browser DOM tree.

Lets create another component in our angular application

You can create a component named user.component.ts which will have name & age properties in its class.

A class must have @Component({}) decorator to behave like a component, the @Component({}) must specify the selector & templateUrl,

selector: tag name of the component

templateUrl: content for the selector

You can use the selector in any other components either in root component or some other components also

@Component: It is a decorator used in angular to make class to behave like a UI component

Decorator: It is a name prefixed with @, it adds extra feature to the class, variables that helps them to have extra capability in the application, there lot of inbuilt decorators in angular which are used on top of the class / variables

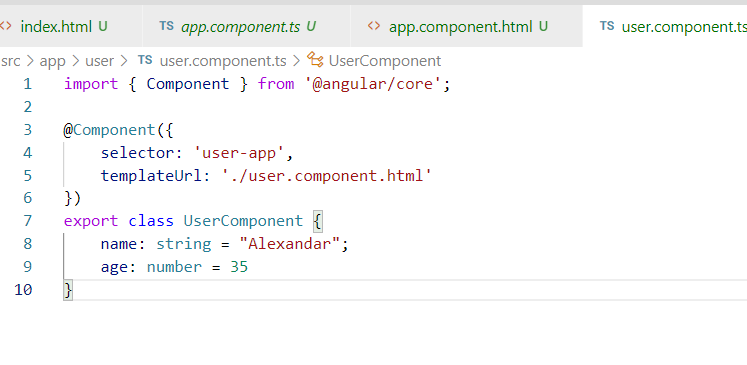
If a class App{} is created it is just a normal class

If a class App{} is created by having @Component({}) on top of it, then it becomes component class, so that the class can share the data to the user and get the data from the user

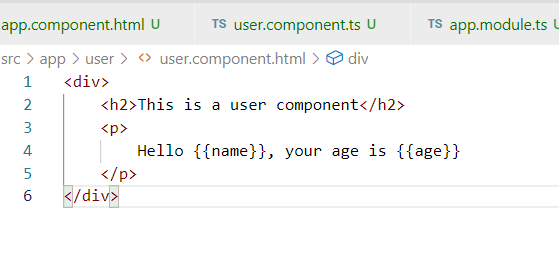
Creating a user component inside app/user folder

Here the ‘user’ folder would have all the files related to user components like user.component.ts, user.component.html, user.component.css

src/app/user/user.component.ts

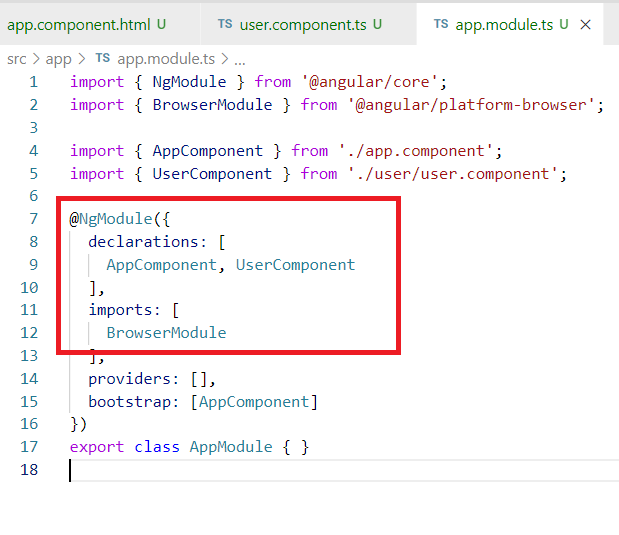


src/app/user/user.component.html



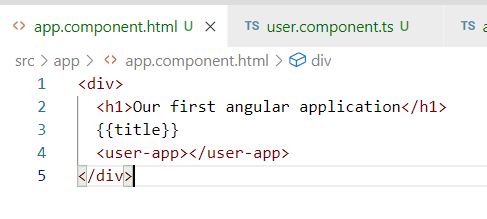
Register the UserComponent in the AppModule

src/app/app.module.ts

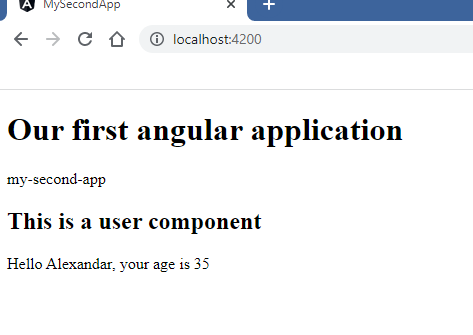


Add the user component in any other components which is part of the root component, since we have only two component here i.e., UserComponent and AppComponent we are adding UserComponent in AppComponent, but it must be added by tag names i.e., selector i.e., <user-app>

src/app/app.component.html



Output:



Templates for the components

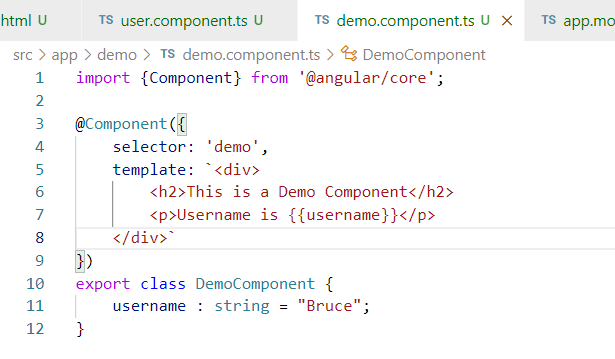
These are the contents written for the component selector (tags), it can be written externally using templateUrl or internally using template which is treated as inline template

@Component({  
 selector:’tag-name’,  
 template: ‘content’  
})

The above component uses inline template, this you can use when the component has only few content, but still you can use external template to get the advantages of the auto-suggestions.

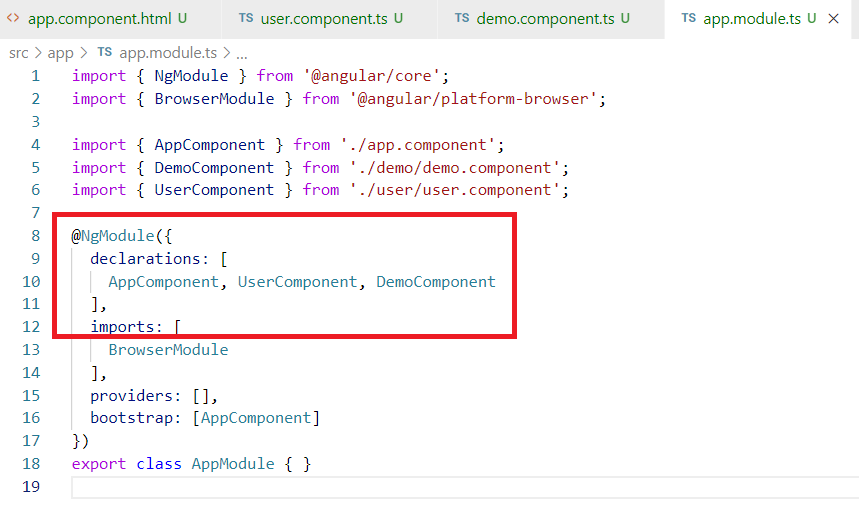
Inline template example

app/demo/demo.component.ts



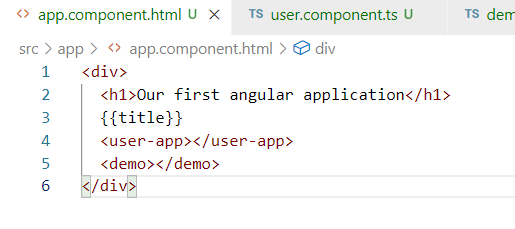
template is used to create an inline content for the component, but it is suitable only when you have a very few lines of contents, if the component contents are more, then you can use templateUrl

Register the DemoComponent in the AppModule app/app.modoule.ts



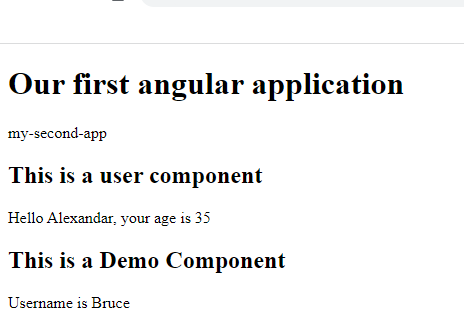
Add the DemoComponent in root component

app/app.component.html



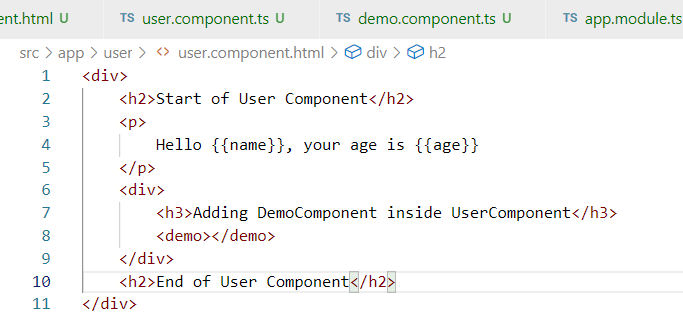
Note: You can add your components in other components also apart from the root component, i.e., you can add demo component int the user component if required

Output:



Let us add <demo> component in the UserComponent i.e., user.component.html

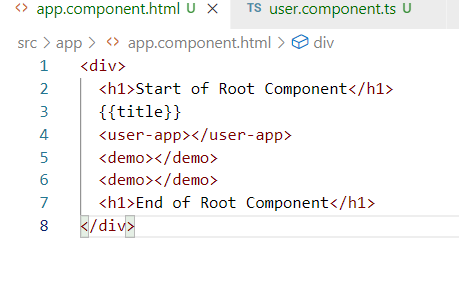
user.component.html



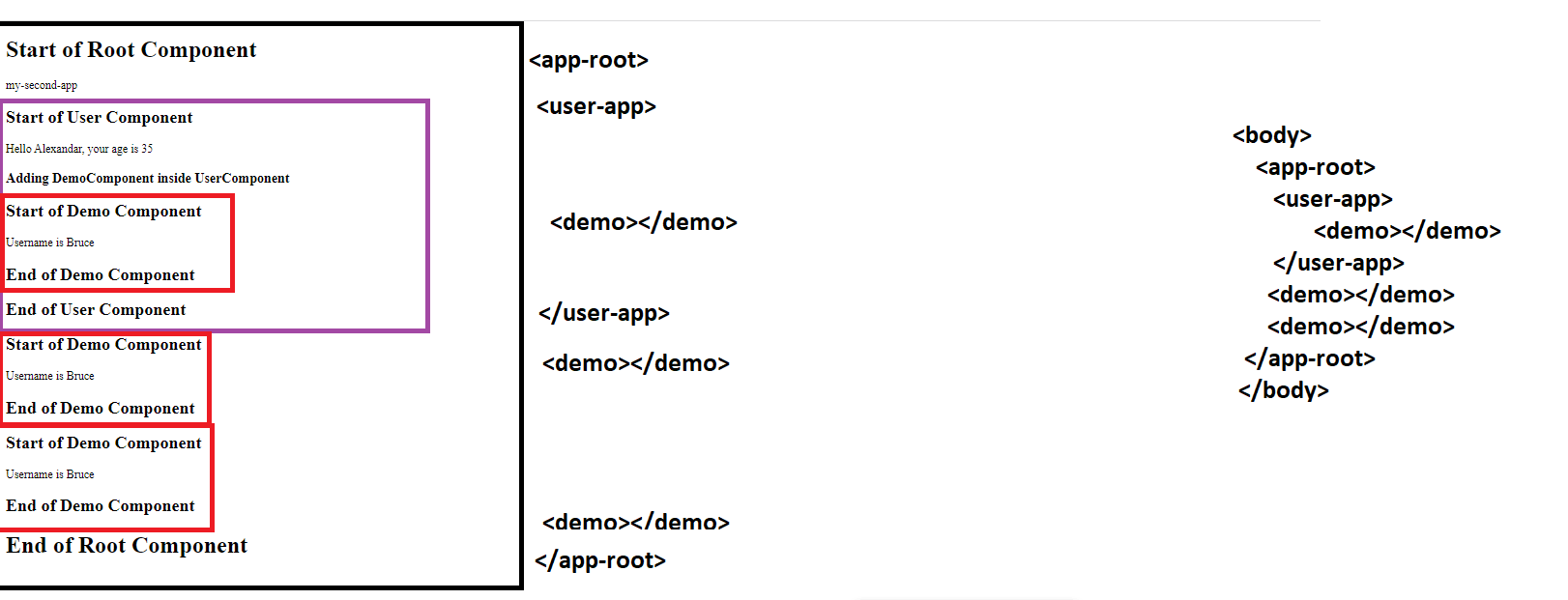
demo.component.ts



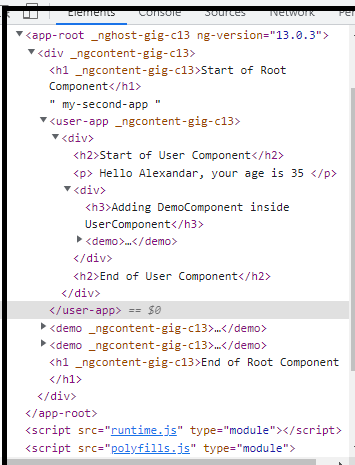
app.component.html



Output:



Browser DOM tree structure

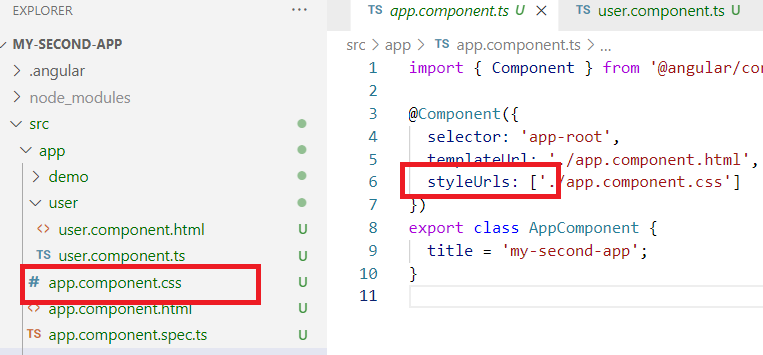


How to style these components

For every components there is a local CSS which applies styles only to the component it is bound to, you can also use global CSS which applies styles to all the components.

Local CSS:

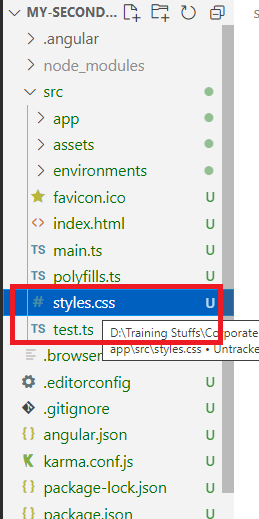
The @Component({styleUrl: [path]}) would mention the Local CSS, where you can apply styles to a particular component



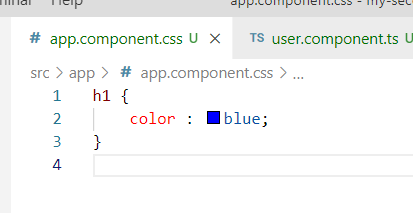
Note: You can create CSS for each component and reference it by using stylesUrls

Global CSS

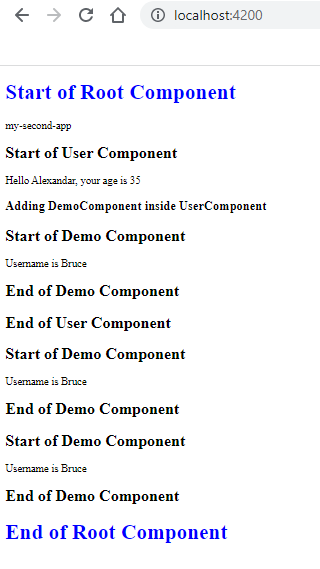
You have a styles.css file in the angular project, that applies CSS to all the components



Let us add style to the app/app.component.css

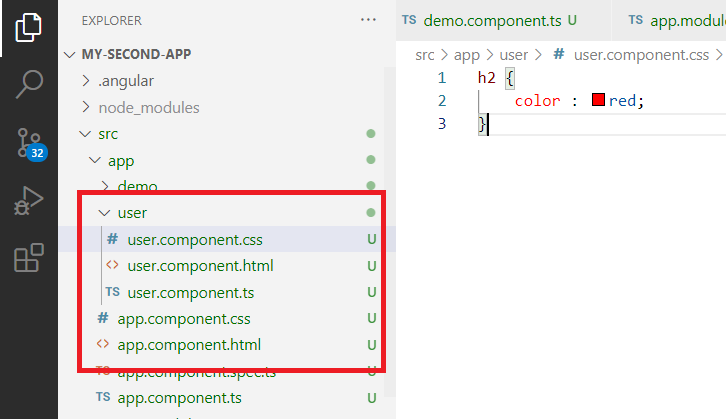


You can see all the <h1> in app.component.html taking blue color

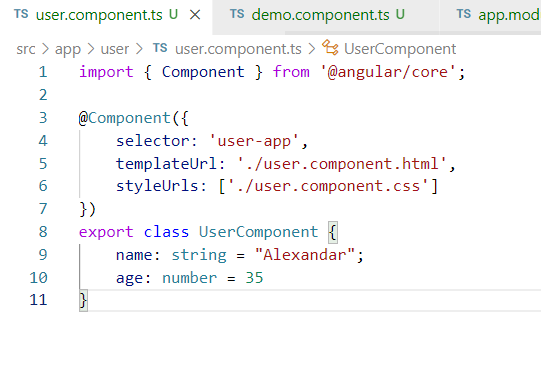


Note: We can create local CSS for the other components

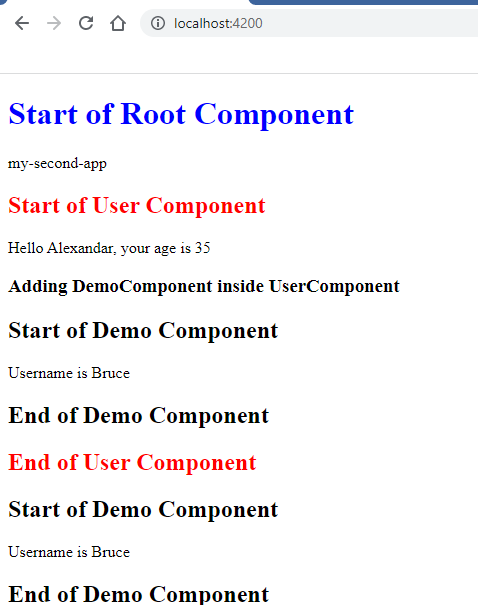
src/app/user/user.component.css



Modify src/app/user/user.component.ts



Output:



Summary:

* ng serve: launches the application by executing main.ts and loading index.html
* AppModule: Root module that represents entire application, this will have all the elements the application should recognize
* AppComponent: Root component which is parent to all the other components
* After creating the component you need to register in the app.module.ts i.e., in the declarations of @NgModule({})
* Inline Template: It is used when the component itself wants to have the content
* styleUrls: It is for adding styles to the particular component, it is a Local CSS
* styles.css: It is for adding styles globally.

Angular CLI commands

Angular provides many inbuilt commands to improve the development speed, these commands are entered with ‘ng’ command

ng generate component component-name (or) ng g c component-name: This is the command which creates .ts, .css, .html files for the components & it also registers the component in the app.module.ts

ex: ng g c demo: This creates 4 files and upates the app.module.ts, those 4 files are

1. demo.component.ts
2. demo.component.css
3. demo.component.html
4. demo.component.spec.ts

Apart from creating these files it will also add some code in some files that developers repeatedly write.

ie..,

@Component({  
 …

})   
export class DemoComponent { }

Apart from ng g c we have many useful commands like

ng generate service service-name (or) ng g s service-name

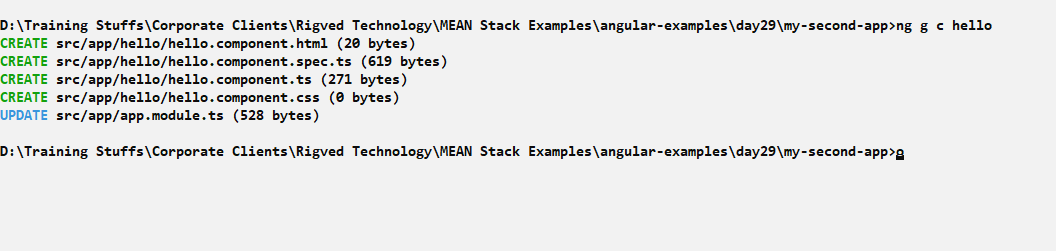
ng generate pipe pipe-name (or) ng g p pipe-name

ng generate guard guard-name (or) ng g g guard-name

All the above commands we are going to see in future

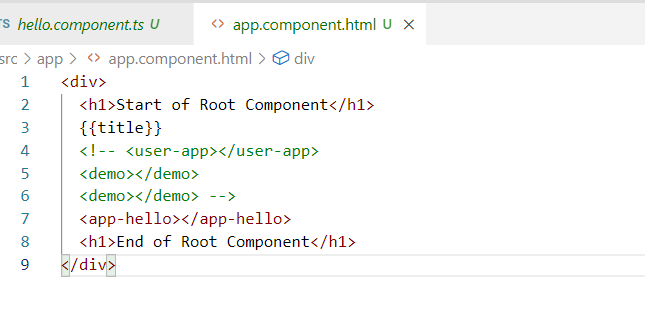
Using ng g c command

To enter this command you must have a separate command prompt that is inside the project folder.

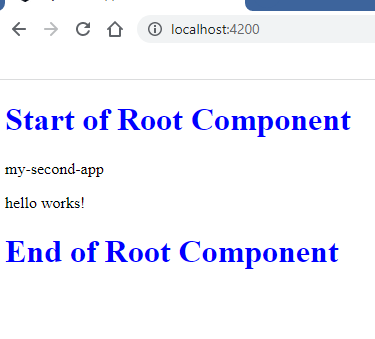


The component is ready to use in other components, you can add in the app.component.html to see the default content i.e., hello works

src/app/app.component.html



Output:



Day 30:

ng serve --port 4201

The above command runs the application in 4201 port instead of 4200

Command to generate component

ng g c component-name

The above command creates templateUrl to represent the view outside the component class

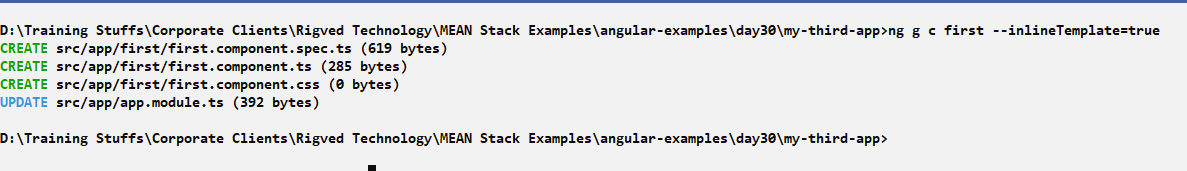
@Component({  
 selector : ‘tag-name’,  
 templateUrl: ‘url’  
})

Command to generate component with inline template

ng g c component-name --inlineTemplate=true

The above command generates the component and view will be part of the component class as below

@Component({  
 selector : ‘tag-name’,   
 template: ‘<p>component wroks</p>’  
 styleUrls: []  
})

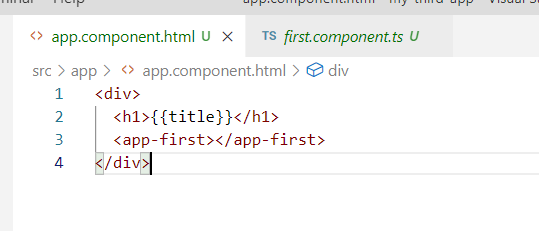


first.component.ts

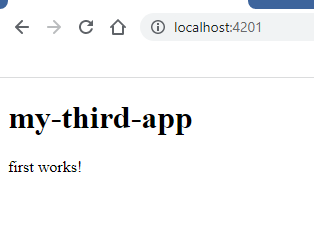


To see the first component content you must add its selector/tag-name in the root component

app.component.html



Output:



Angular Pipes(|)

These are the features in the angular which can format the output so that the content will be transformed to something else instead showing the original content, angular provides lot of inbuilt pipes that can use in the component template using pipe symbol(|), the inbuilt pipes are

uppercase, lowercase, currency, date, json

{{data | uppercase}}: The data will be shown in uppercase

{{data|lowercase}}: The data will be shown in lowercase

{{data|currency}}: This data will show the number in currency format, by default it will be $, but you can pass arguments to the pipes if you need some other currency symbol

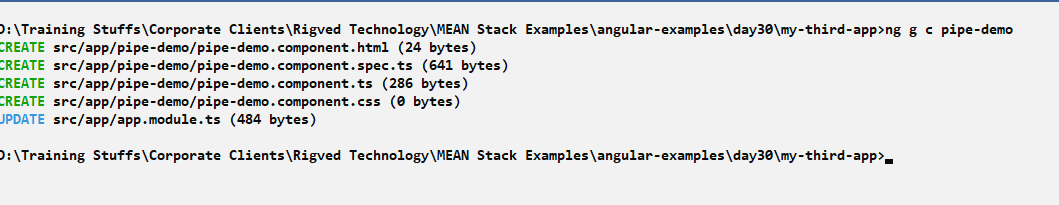
{{data | currency : ‘INR’}}: Here the currency is the pipe name & INR is the argument that specifies the condition what the currency should be in, it will show rupee symbol

{{data | date}}: Shows data in default date format, but it can be formatted according to the requirement

{{data | date: ‘MM-dd-yyyy’}}: Date will be in MM-dd-yyyy format

{{data | json}}: Converts the javascript object to json and shows that json to the user

Note: The data must be of right type when we are formatting, else it doesn’t work, ex: If currency is applied then it must be applied on number type data only.

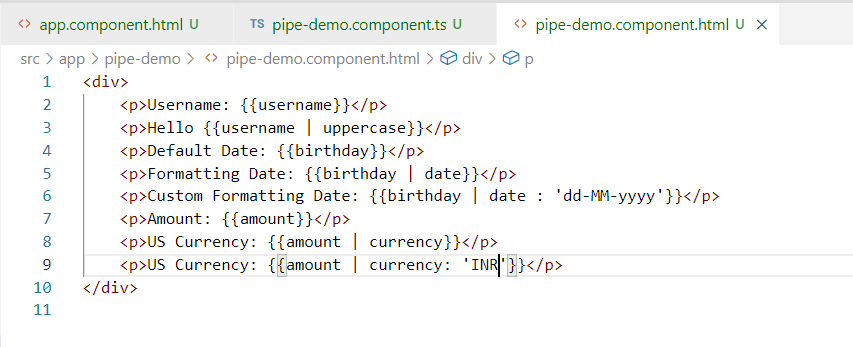


pipe-demo.component.ts

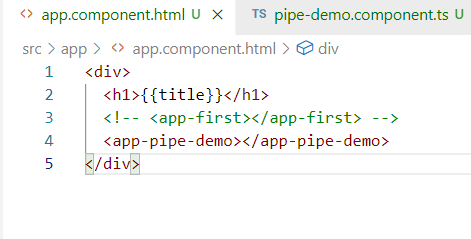


We have username, birthday, amount in the component class that can be formatted using the inbuilt pipes

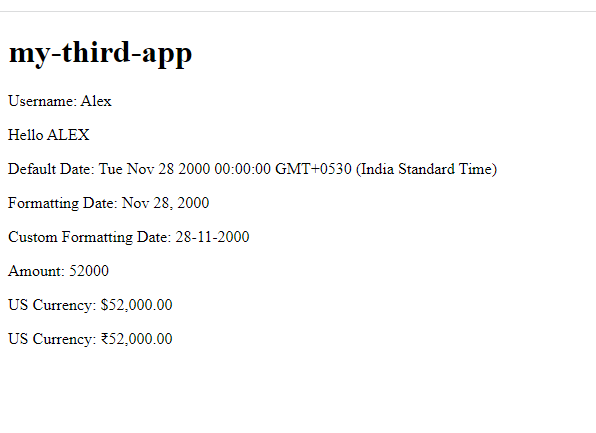
pipe-demo.component.html



Note: Add this component to the root component



Output:



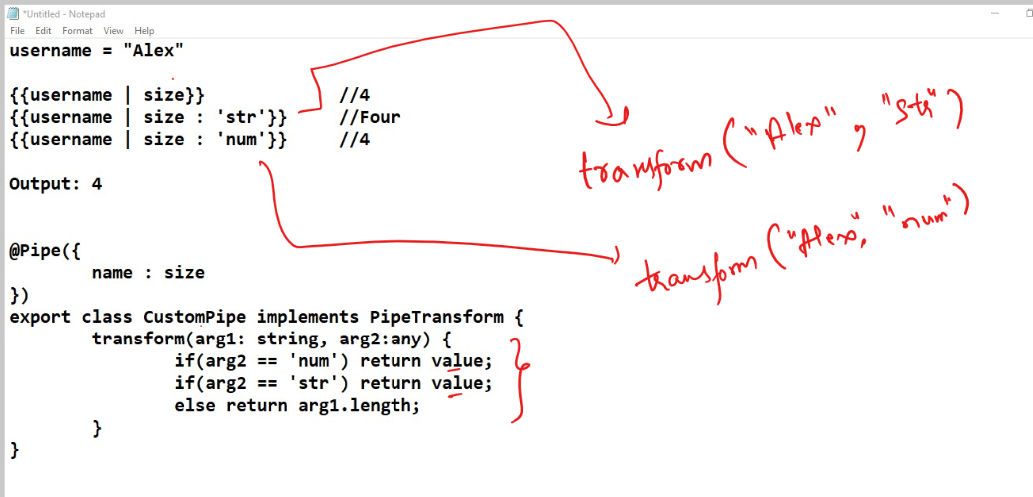
Custom pipes:

These are created when you need your own pipes for transforming the data when the predefined pipes doesn’t help your requirement

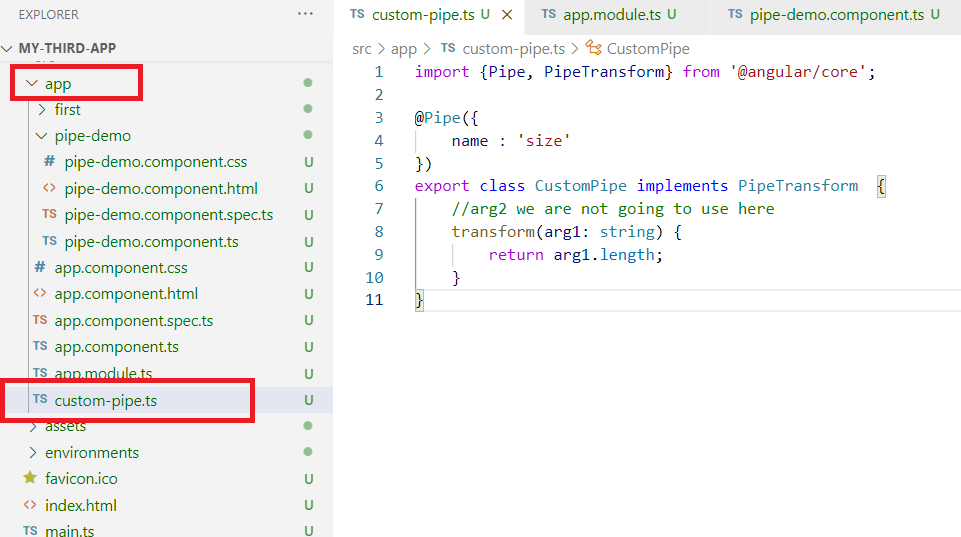
PipeTransform: It is an interface a class must implement while creating the custom pipe, the PipeTransform has a function called transform() that is executed when the custom pipe name is used, the data transform returns would be the transformed data that comes after applying the pipe

@Pipe: It is used to decorate a class to act like a pipe, it will have a name which is used while transforming the output,

Note: A class that implements PipeTransform must have @Pipe decorator, it must be registered in the AppModule so that angular calls the transform() function at runtime automatically by passing arguments to the transform() whenever the pipe is used

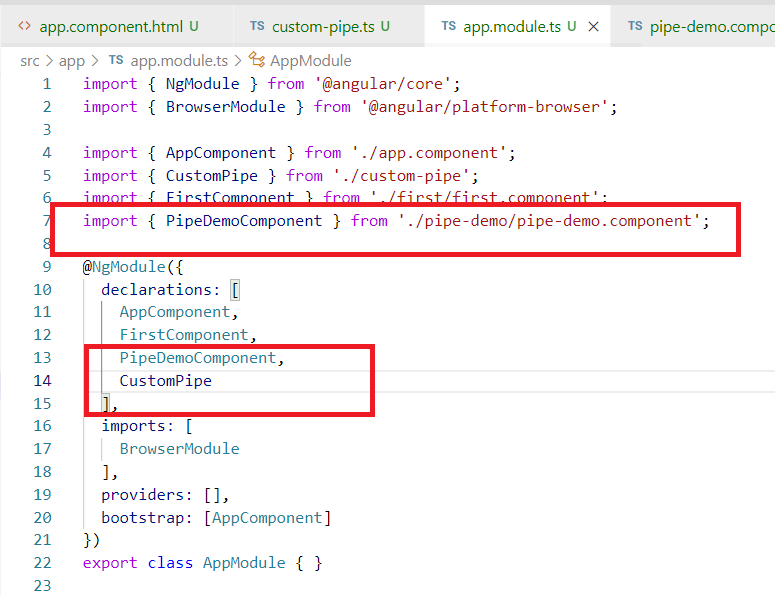


src/app/custom-pipe.ts



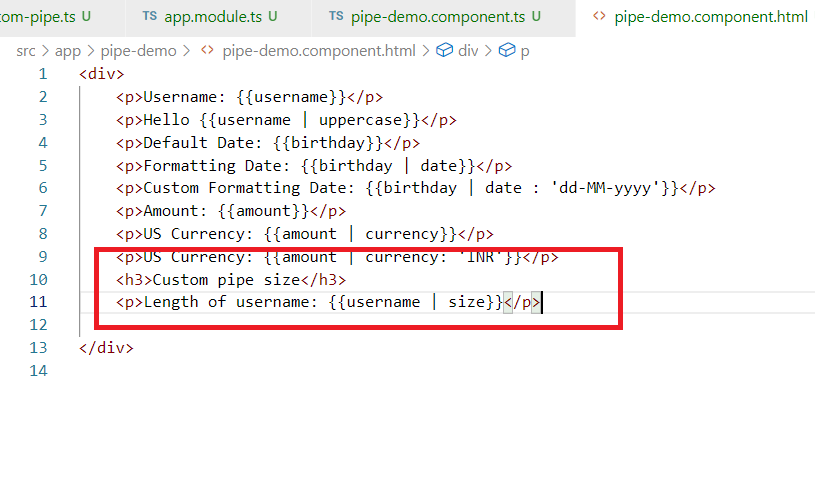
The ‘size’ is the pipe name which needs to be used when you want to transform, but angular needs to know this pipe name hence you must register in app.module.ts

app.module.ts



Now you can use the pipe name size anywhere you want and see the size of the string

pipe-demo.component.html



Output:

