**Install nodejs**

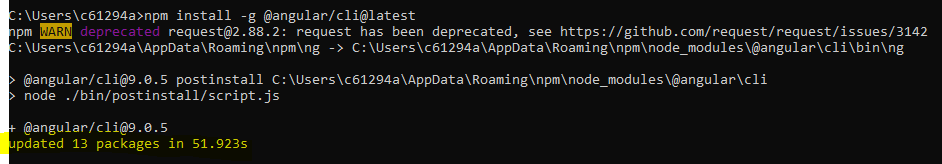
Download latest version of nodejs from nodejs website and install the application

**Angular CLI installation**

Run below command in the terminal

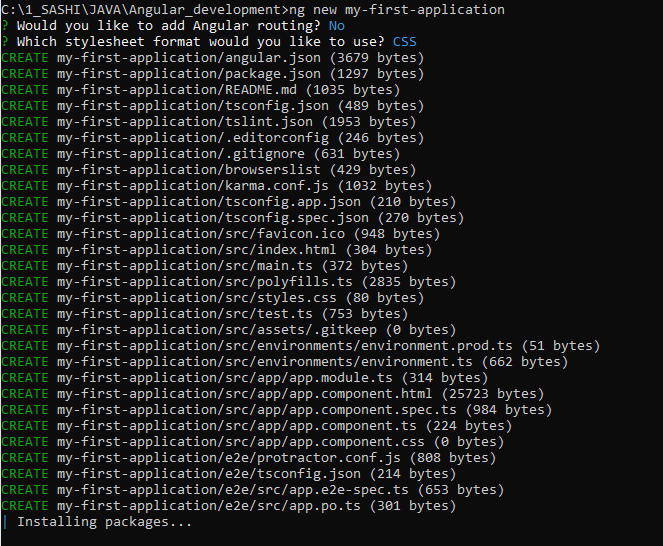
npm install -g @angular/cli@latest

here -g : globally – Installing the angular cli globally



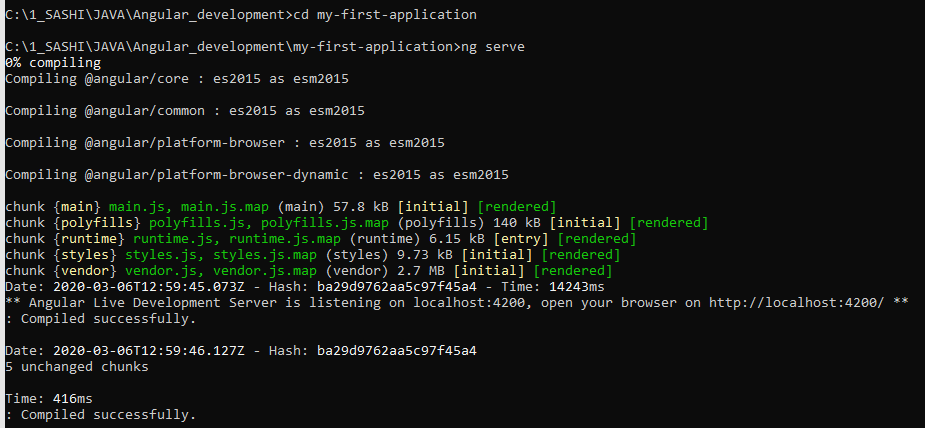
**Creating the first application**

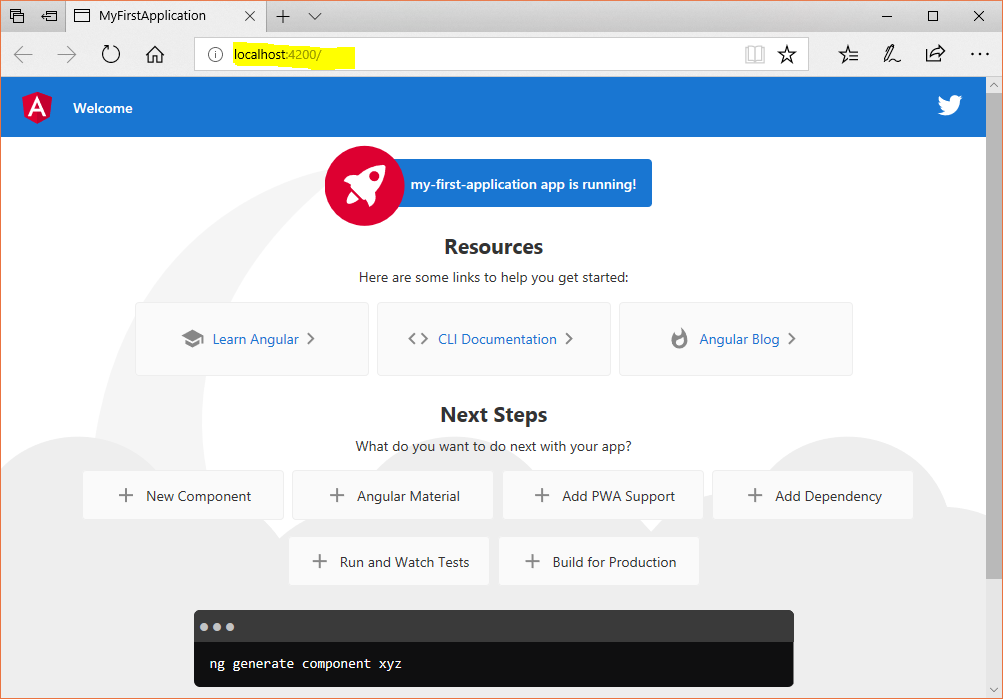
ng new <app\_name>



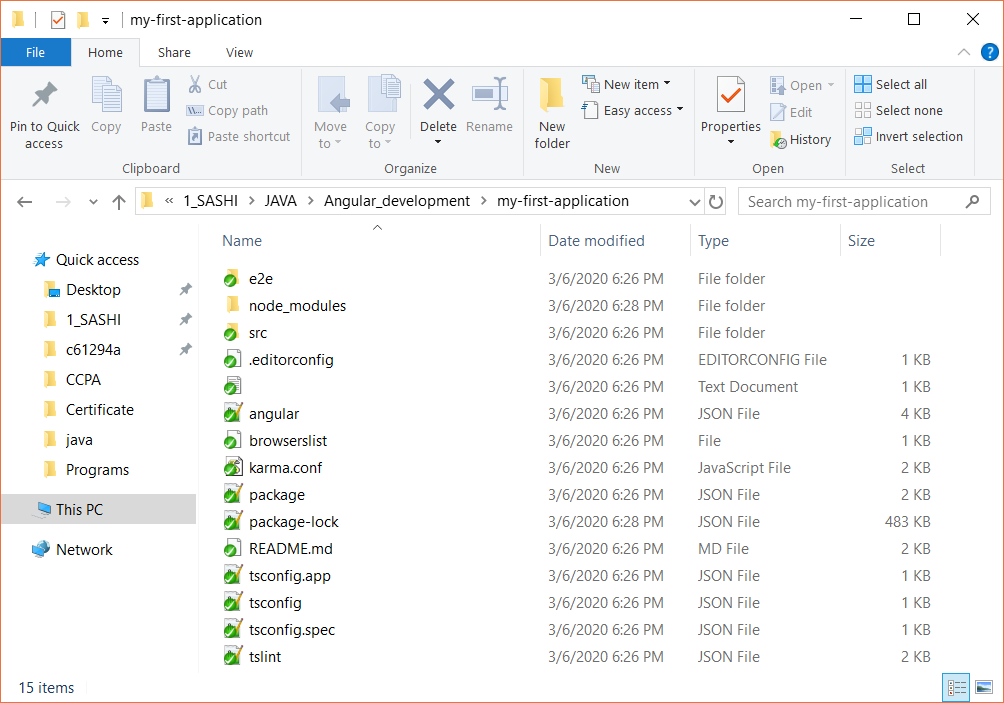
**Running the first application**

ng serve



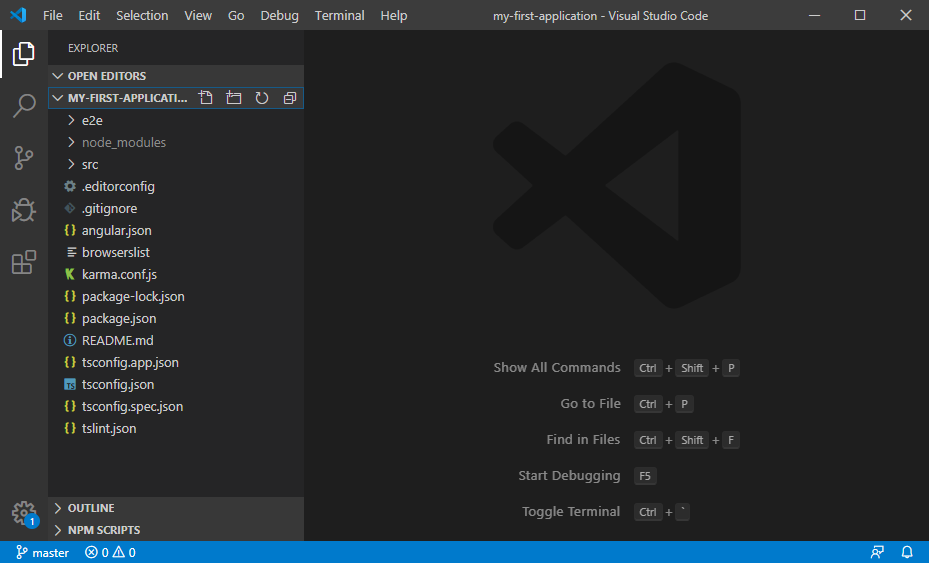


Structure of the project



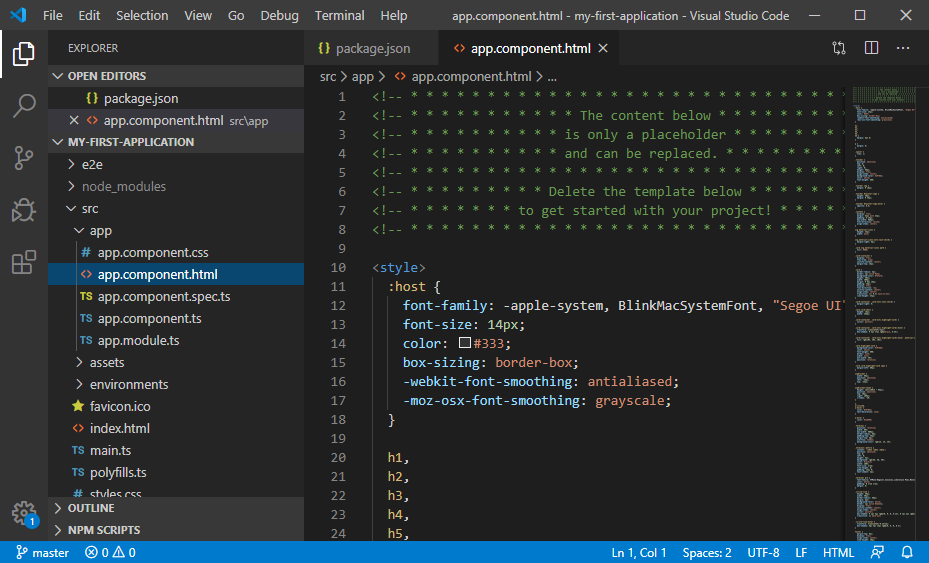
Download visual studio code from internet

Open the first project in the visual studio code editor

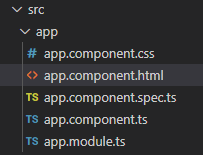


package.json – contains all the dependencies

Open the src 🡪 app 🡪 app.component.html file. This is the page, which is displayed, while we run the application.



app has below things



Change something in the

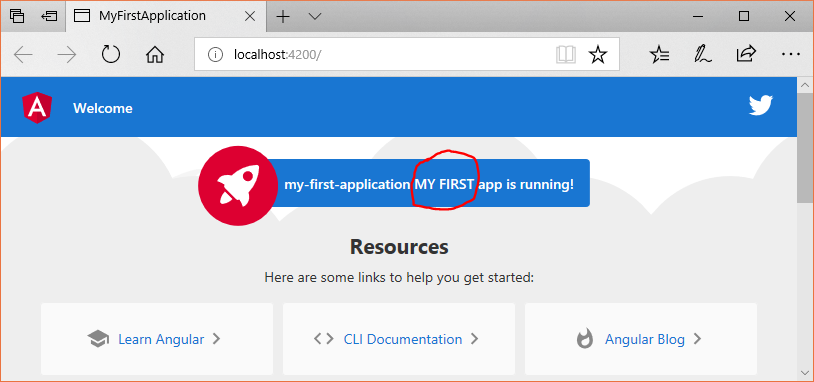
The below is the default one

<span>{{ title }} app is running!</span>

Change as

<span>{{ title }} MY FIRST app is running!</span>

Save it. No need to re-start the server. Look into the console.



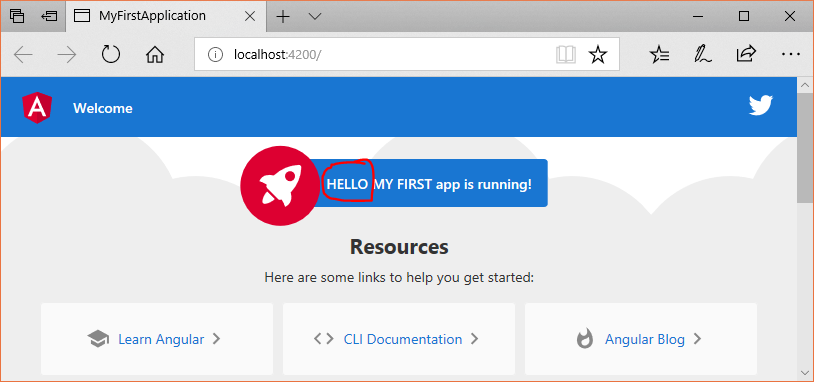
We will change the title

Open 🡪 app.component.ts 🡪 change the title as below

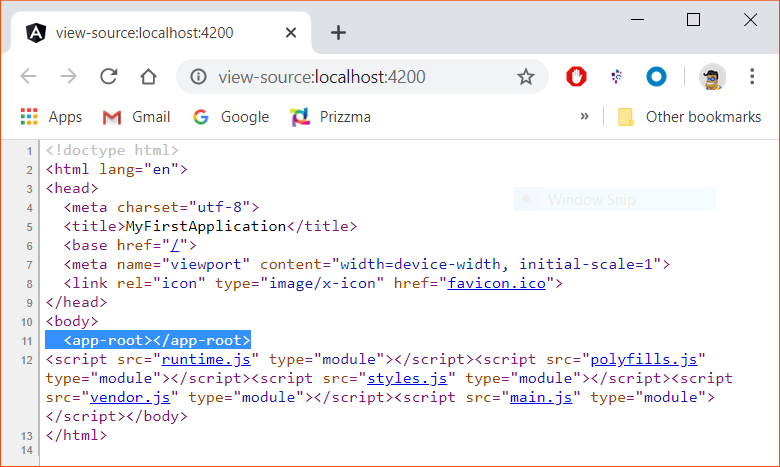
export class AppComponent {

  title = 'HELLO';

}



Inspect the element of this web page

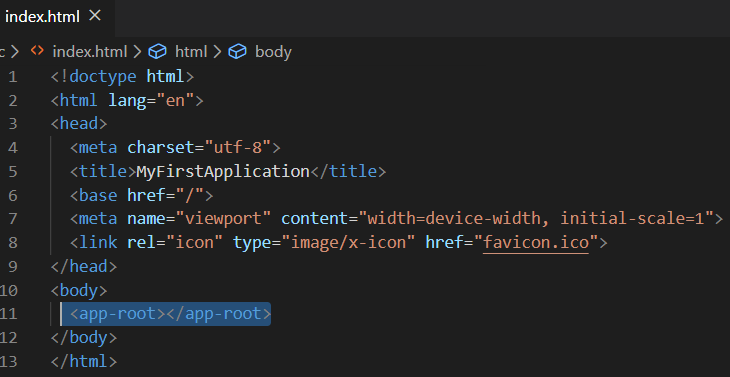


We can see, under body section, <app-root> is the complete kind of html file which takes care of the page.

The same tag <app-root> can be found in app.component.ts file

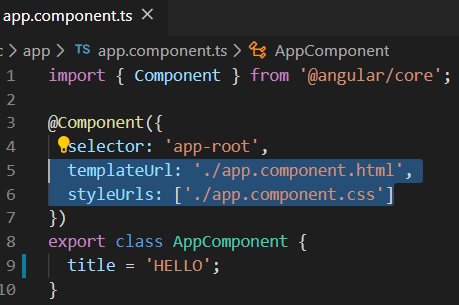


This <app-root> mentioned in the index.html file



How it works?

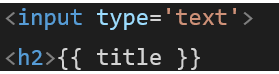
Angular always loads index.html file, while it loads the **index.html** file, it sees <app-root> under body section, <app-root> dynamically gets replaced by **app.component.ts** file (below image). In app.component.ts file, under **selector: 'app-root',**  we mentioned the html file (app.component.html) & css file (app.component.css). Both are picked and shown in index.html file



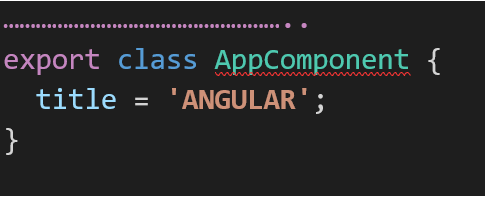
Let’s experiment below

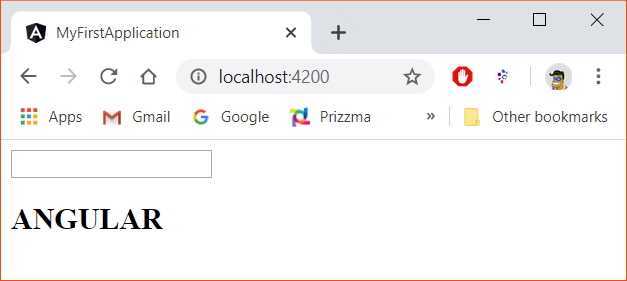
Ex 1 :

Remove everything from app.component.html file and keep below



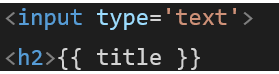
Change the title from app.component.ts file as below





How it works?

Angular loads index.html file, while it loads the **index.html** file, it sees <app-root> under body section, <app-root> dynamically gets replaced by **app.component.ts** file (below image). In app.component.ts file, under **selector: 'app-root',**  we mentioned the html file (app.component.html). In app.component.html file we mentioned only below one. So we’re getting above result



**ngModuel directive**

whatever we’re typing in the text box, same we can display using ngModel.

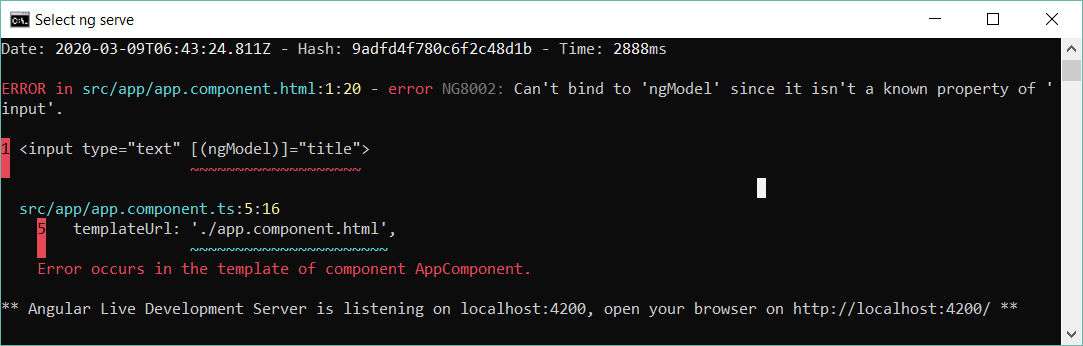
For that change in app.component.html file as shown below

<input type='text' [(ngModel)]="title">

<h2>{{ title }}

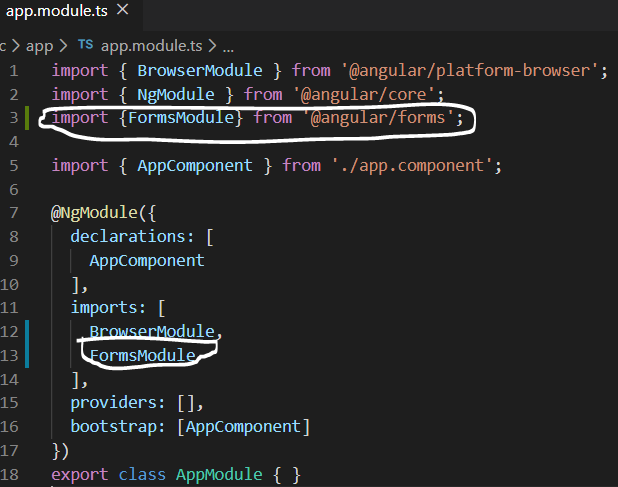
Whatever you are input in the text box, same copied to title label.

While running, getting below error.

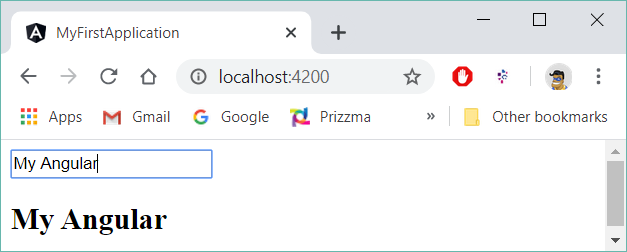


We need to import ngModel , because ngModel is not from Angular, it is from type script.

In app.module.ts, we can mention the import statement for ngModel



Now save it and see below



What is type script?



**Creating the second application with Bootstrap styling**

ng new my-second-app

import the project into vs code

To install bootstrap latest version follow below:

1. install the bootstrap latest version in the current project. This bootstrap applicable only projectwise.

C:\1\_SASHI\angular\my-second-app>**npm install --save bootstrap@4**

Here bootstrap version is 4. And it is only available to my-second-app project only.

1. To aware Angular to use the downloaded bootstrap, go to **angular.json** file.

Already one style file is mentioned as below

"styles": [

"src/styles.css"

],

You can navigate to src 🡪 style.css file. In this file, the global css can be defined.

Here, we can defined the node\_modules under node\_modules/bootstrap/dist/css/bootstrap.min.css file as shown below

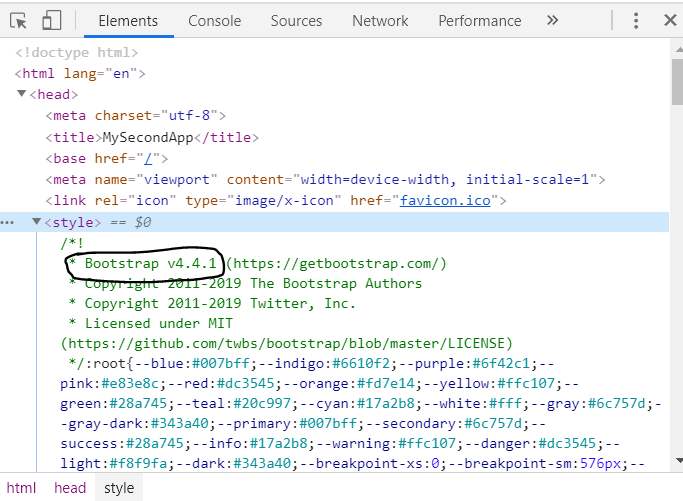
"styles": [

**"node\_modules/bootstrap/dist/css/bootstrap.min.css",**

"src/styles.css"

],

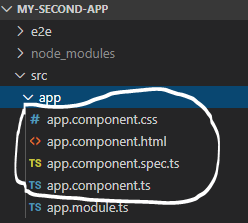
Run the application and check in terminal, go to developer tools 🡪 Elements Tab 🡪 Under <head> section, we can see it is using bootstrap 4



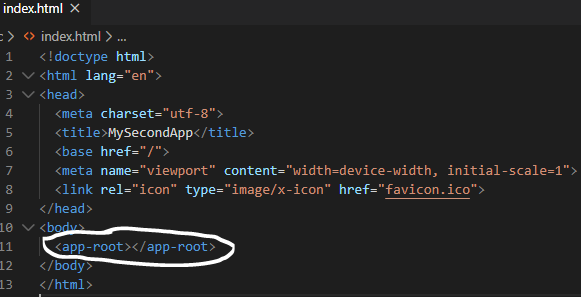
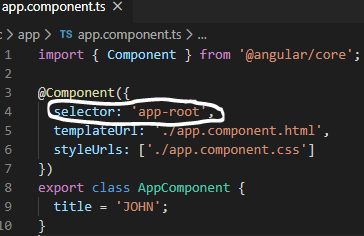
**Diving deeper – How an Angular app gets loaded and started? Or**

**What happens when we visit localhost:4200 ?**

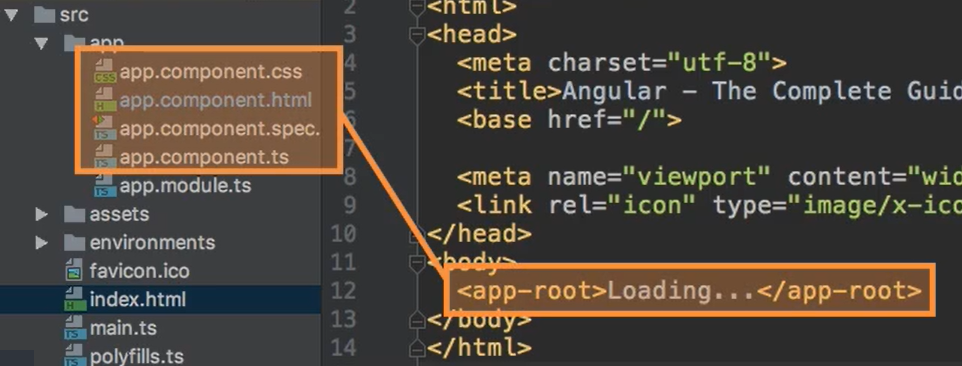
The below files are belong to one component called app



In index.html file, under body section, <app-root> section is there, same it is mentioned in the app.component.ts file under selector.

So Angular replacing the <app-root> section with above component elements. And this app component contains the app.component.html file, so it loads that file



How Angular triggers the index.html file?

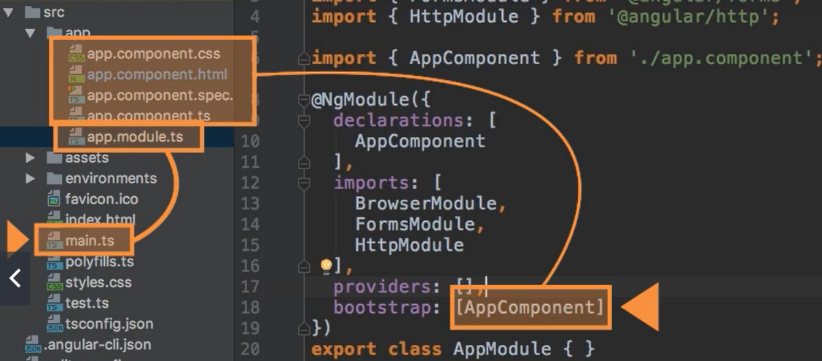
When we run the application using ng serve, first **main.ts** file gets executed. Here, we’re passing **AppModule** to bootstrapModule method

platformBrowserDynamic().bootstrapModule(AppModule)

Using this AppModule, it selects **app.module.ts** file.



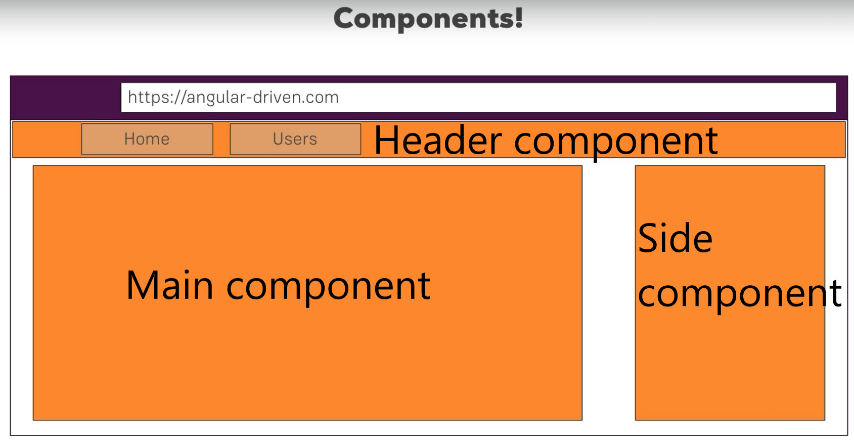
In app.module.ts file, the component is mentioned as **AppComponent.** Ie bootstrap : [AppComponent]. From here, it finds all the files (<app>.component.css, <app>.component.html & so on) from the app component.



Then it checks, <app>.component.ts file and finds the selector name is ‘app-root’. This app-root is used in the index.html file. Now index.html file knows which selecter to use it. So it loads the correct app selecter. Now app.component.html file gets injected and executed.

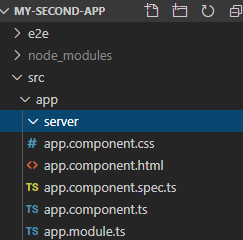
**Component**

A page can have multiple components has shown below, we can inject many components.

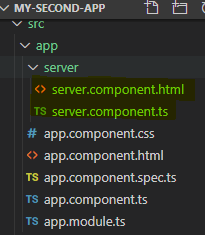


**Creating a new Component**

1. New component should be created inside src 🡪 app directory 🡪 server (folder)



Create server.component.html & server.component.ts files inside server folder as below

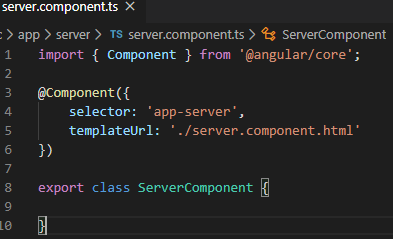


Keep below contents in server.component.html file

<h2>This is server component</h2>

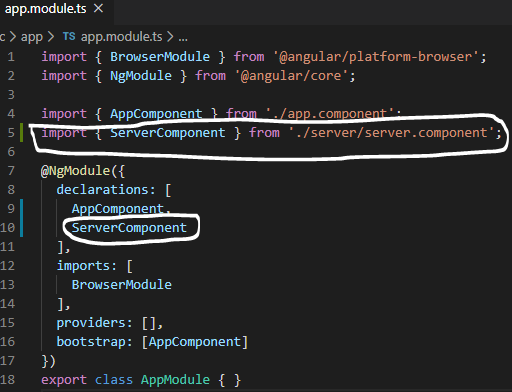
Here, declaring a class called ServerComponent. We mentioned export, it means, this ServerComponent class should be visible to other components.

We’re using @Component decorator to make this class as component. Inside, we’re mentioning a selector called <app-component\_name> ie app\_server. This selector name should be unique.



2. Registering newly created ServerComponent

So far, we’ve just created a component. For Angular to know this component is created, we need to register in **app.module.ts** file. We’re mentioning the newly created component name (ServerComponent) inside ngModule declaration section. And its import should be mentioned. In import statement no need to mention the .ts extention, only Server.component is enough

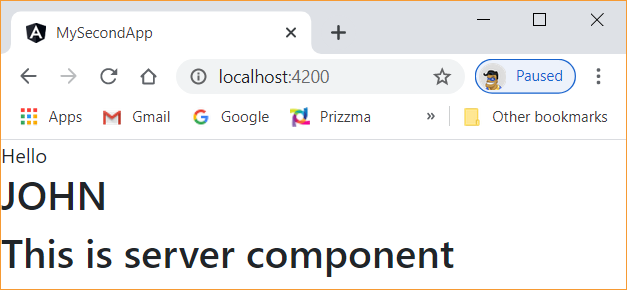


3. Mention the Server component selector name in the app.component.html file.

Hello <h2> {{ title }}</h2>

<app-server></app-server>

Output



**Creating components with CLI**

Create a new application (app3)

ng new app3

Install bootstrap

1. install the bootstrap latest version in the current project. This bootstrap applicable only projectwise.

**npm install --save bootstrap@4**

1. To aware Angular to use the downloaded bootstrap, go to **angular.json** file and add below

"styles": [

**"node\_modules/bootstrap/dist/css/bootstrap.min.css",**

"src/styles.css"

],

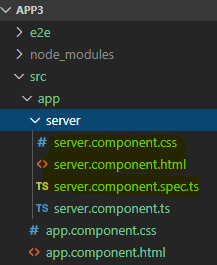
For creating new component :

ng generate component <component\_name> or

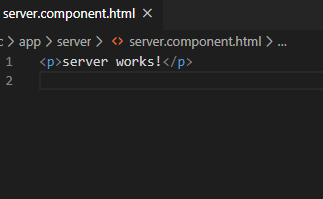
ng g c <component\_name>

**Creating first component :** ng g c server

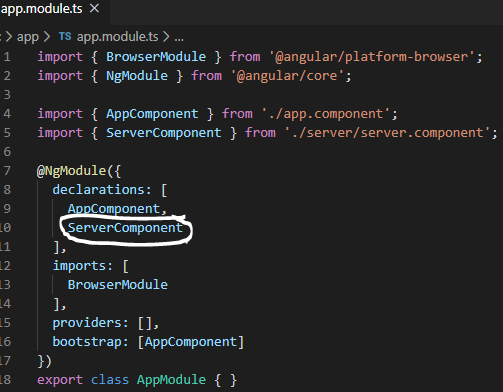
Here server component is created and has all below files



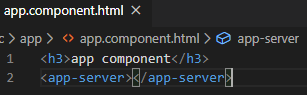
Check app.component.ts & app.component.html files. CLI only created all these files

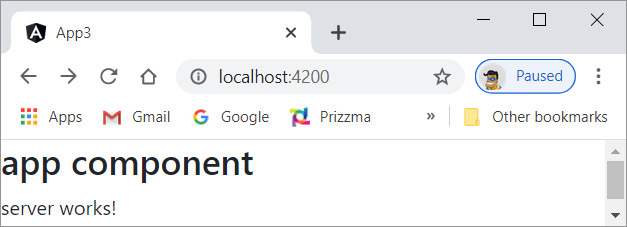
Check app.module.ts file. CLI only injected ServerComponent in this file



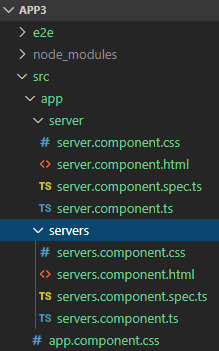
Add server component selector in app.component.html file



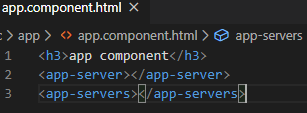
Output :



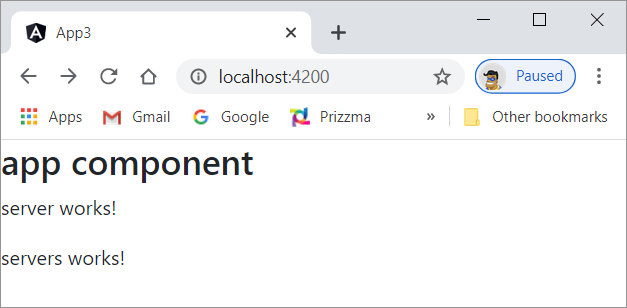
**Creating second component :** ng g c servers



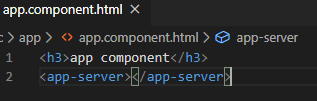
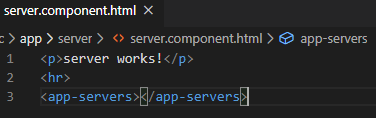
1. Add servers component selector in app.component.html file



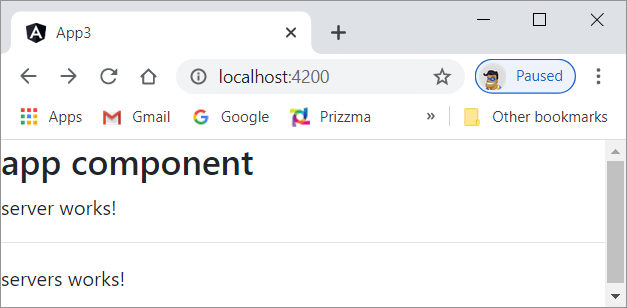
Output :



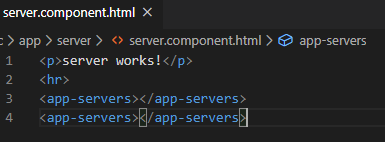
Remove <app-servers> from app.component.html and keep that in server.component.html file

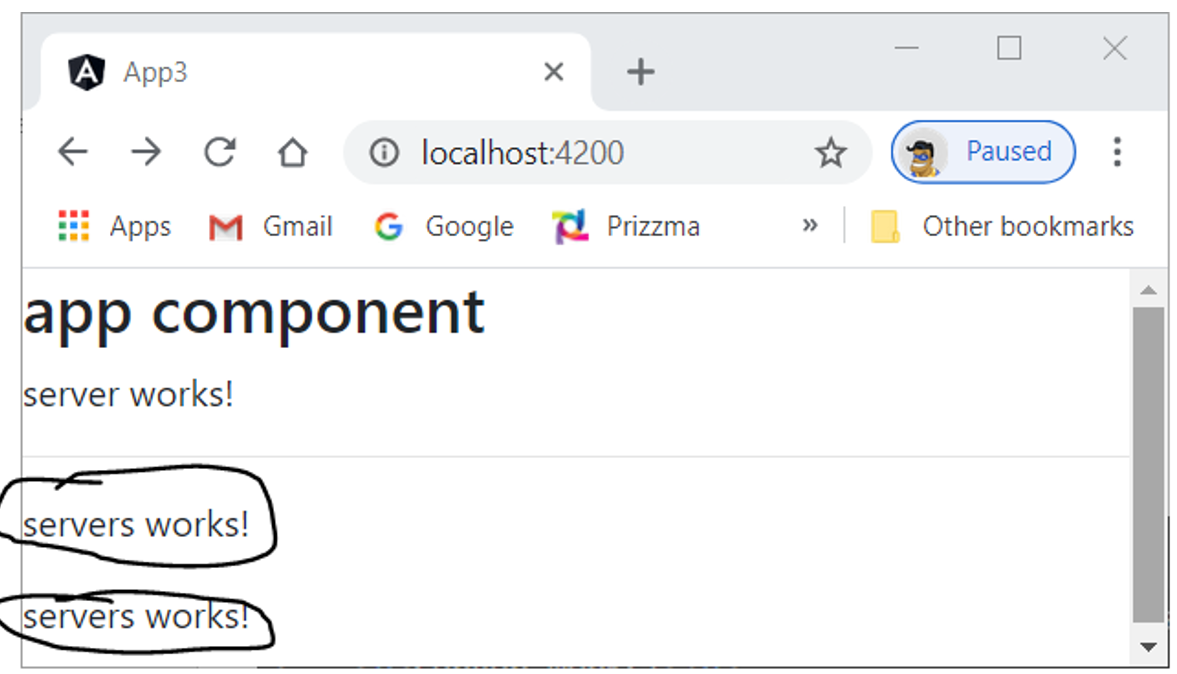
 

Output

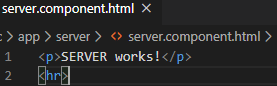
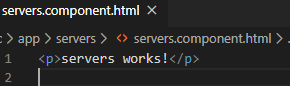


We can re-use the components

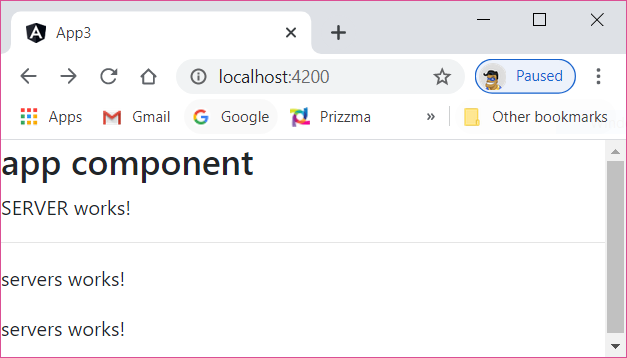




To identify the differences, change below files

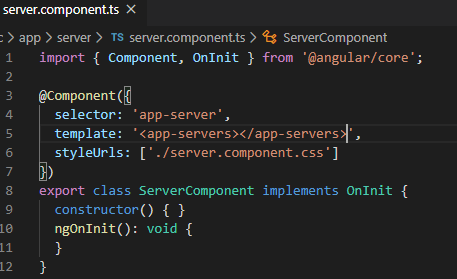
 

Output

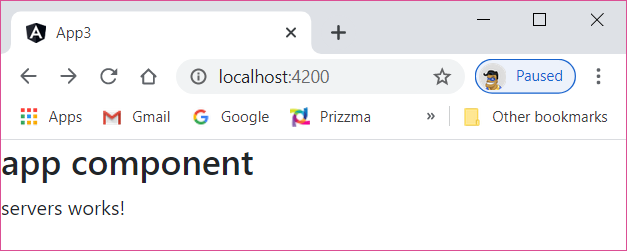


**Working with Component Templates or definding html code in template**

Remove TemplateUrl from server.component.ts file and keep only template and mention the component inside.



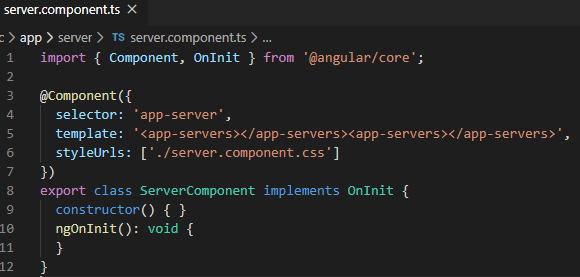
Output

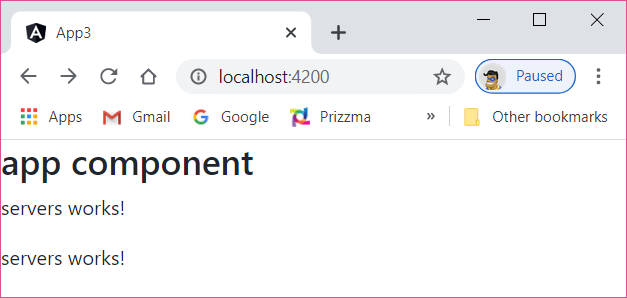


Explanation:

In server.component.ts file, instead of pointing out to server.component.html file, we’re pointing to <app-servers> component. So it goes to appServers component, from there it goes to servers.component.html file.

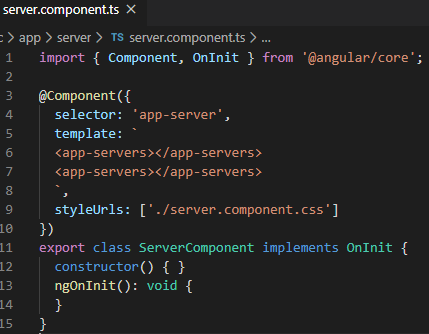
**Adding more than one component in template**

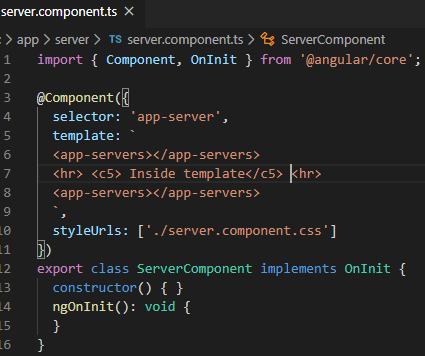


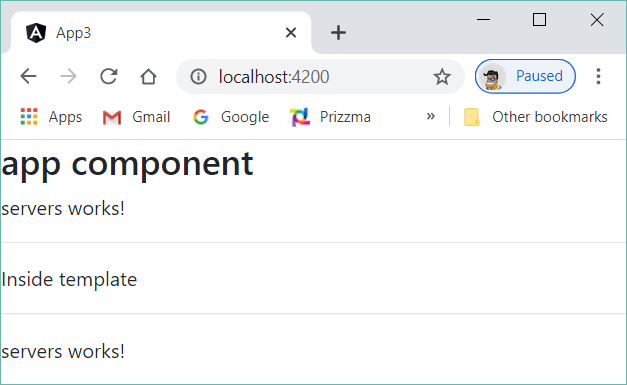


Here, we cannot break the line in template.

If we want to break the line, then we can use ` `







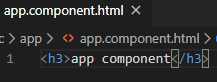
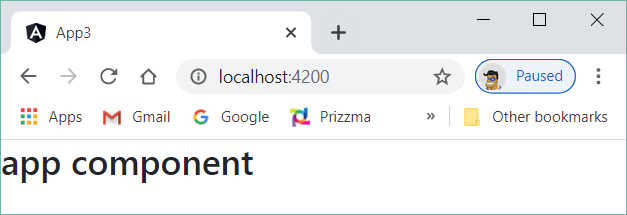
Using Component Styles (CSS)

1. External file

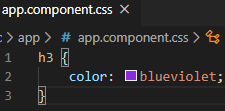
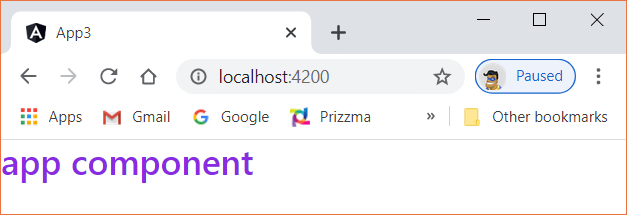
2. Inline style

1. External file

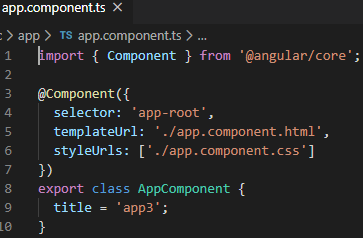
Keep only below.

Keep below in this file

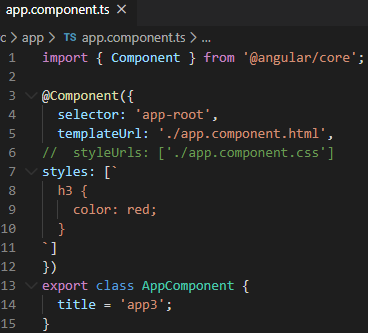
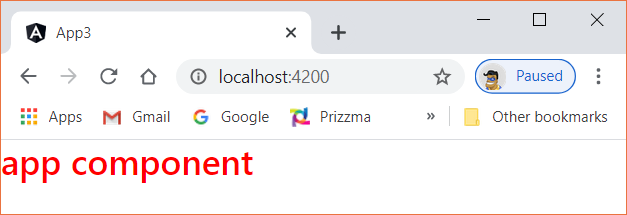
 

Here, app.component.ts file contains StyleUrls : as app.component.css file. Inside that css file, for h3 tag, the color is mentioned. Same style is getting applied.



2. Inline style

Remove the styleUrls and keep the styles which takes array, there mention the css style inline

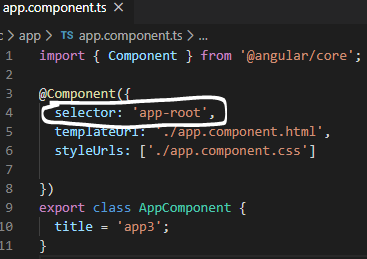
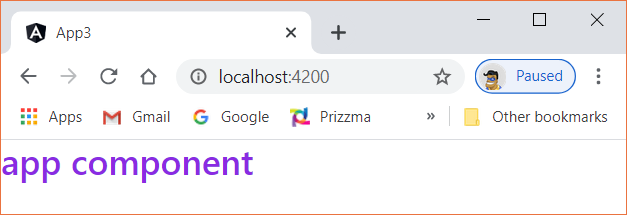
 

Understanding Component Selector

1. Using component name
2. Using div style
3. Using class style

1. Using component name

This we already aware of it. By using the selector : ‘app-root’, it will work.

2. Using div style