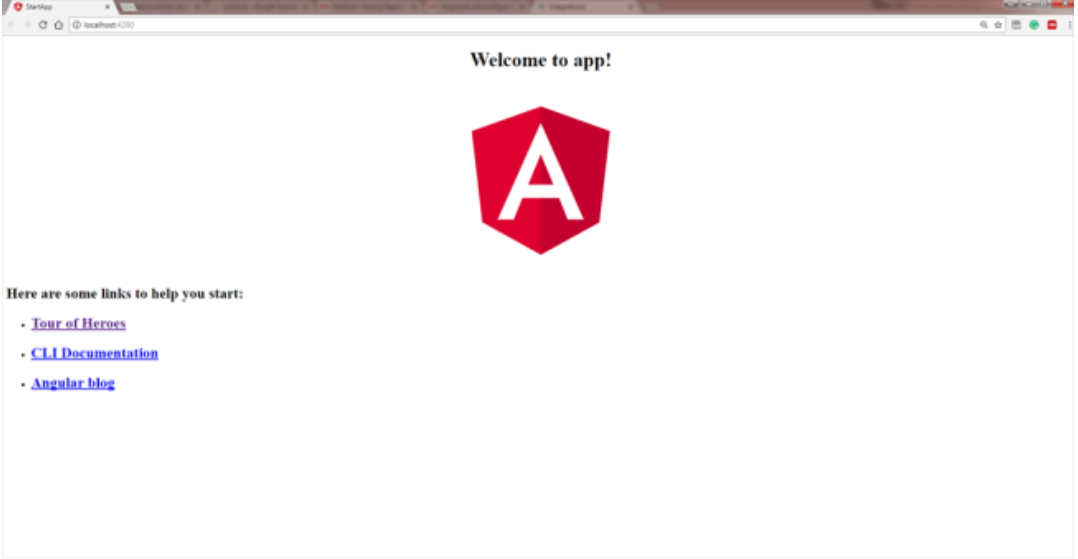


Ref : https://www.c-sharpcorner.com/article/how-an-angular-5-application-get-started-or-loaded/

How An Angular 5 Application Gets Started Or Loaded

Now, we will try to understand how an Angular application is loaded and gets started.

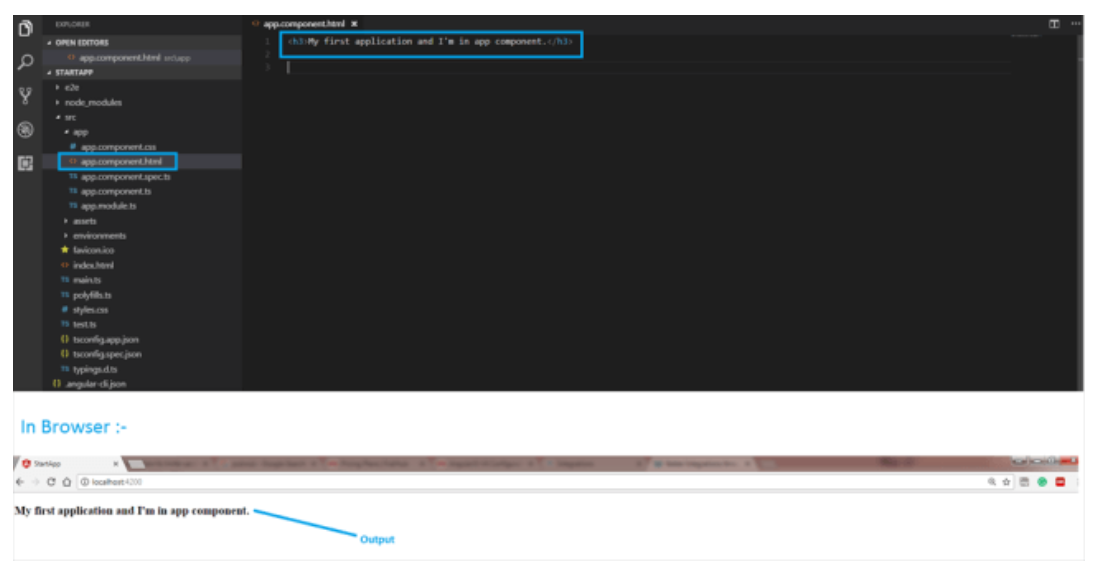
Whenever we create an Angular project and run it with the help of ng-serve command, it looks like the below image.



Now, we will try to understand *how an Angular application is loaded and started.* For this, we need to remove all the code from the app.compoment.html file and write some basic HTML code. Something like below.

1. <h3>My first application and I'm in app component.</h3>

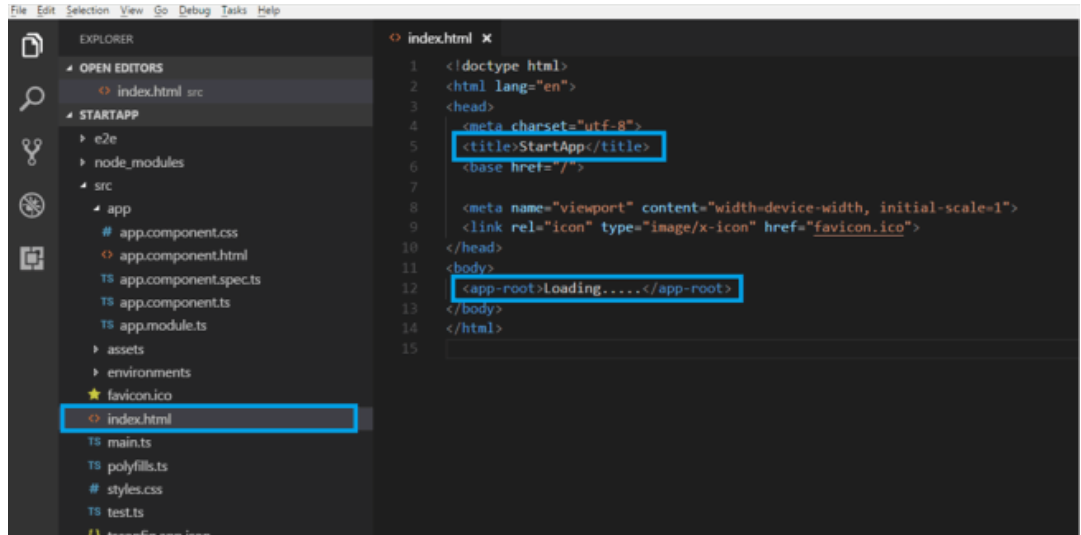
Now, our application looks like this...

  
  
  
But the concept is that this file is not the file served by the server. The server served an *index.html* file. Remember that Angular is a framework which allows us to create *“Single Page Applications”*, and it (index.html) is the single page which was provided by the server.

In index.html file,

1. <html lang="en">
2. <head>
3. <meta charset="utf-8">
4. <title>StartApp</title>
5. <base href="/">
6. <meta name="viewport" content="width=device-width, initial-scale=1">
7. <link rel="icon" type="image/x-icon" href="favicon.ico">
8. </head>
9. <body>
10. <app-root>Loading.....</app-root>
11. </body>
12. </html>

We look into this file,

  
  
  
It is a normal HTML file as here, we can see the “title” tag and we can see the same title in the browser as our application title.

If we talk about the “body” tag, it is a little bit different from our normal HTML page's body. Here, we can see “<app-root>” tag, and it does not reflect a default tag. It contains “loading...” as a text, but we have never seen this on a page, which means somehow, the page has been changed.

So, here, we come to know that it is a component which is provided by the CLI. We can say that, whenever we create a project from CLI, by default, one component is created, i.e., “app component”.

Now, look into “app.component.ts” file. It is a TypeScript file. We will discuss the component and how we can create a custom component in my next blog. Here, the main thing is the “selector” property.

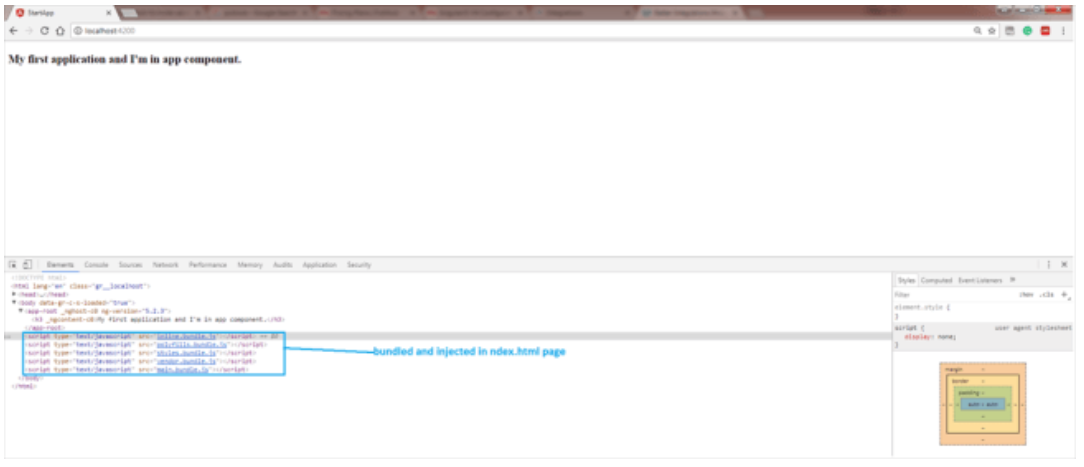
In app.component.ts file, the code is as below.

1. **import** { Component } from '@angular/core';
3. @Component({
4. selector: 'app-root',
5. templateUrl: './app.component.html',
6. styleUrls: ['./app.component.css']
7. })
9. **export** **class** AppComponent {
10. title = 'app';
11. }

Now, look at the relationship between these two files.  
  
   
  
Selector property contains exactly the same string as we have in our index.html file. This information is needed by the Angular to place this part into an index.html file with the template of the component.

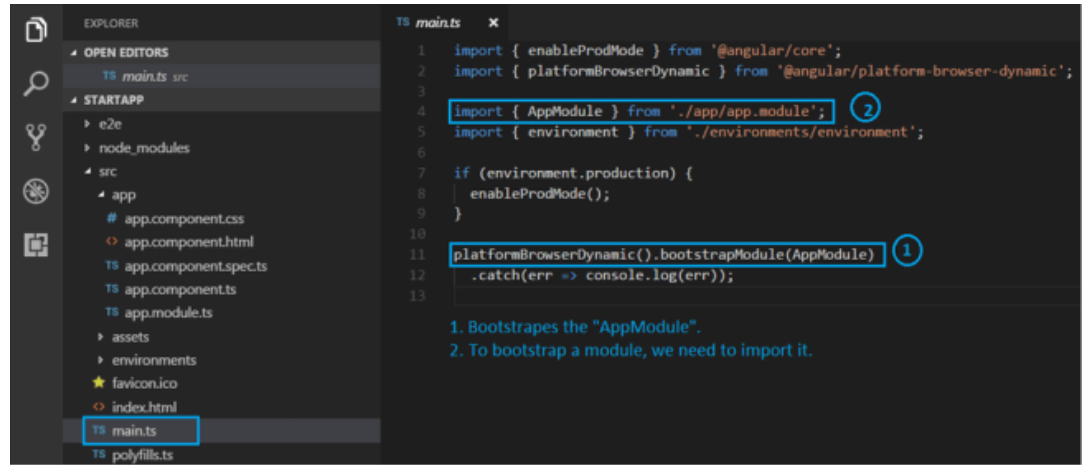
The template of the component is “./app.component.html”, so, Angular includes this part into the body of the index.html file.

Now, we are able to understand how an “app-root” is included in our index.html file. But here, the main question is “How did Angular trigger or how did it kick off ?”

Let us understand this concept. We need to inspect our Angular application and we may find a couple of script files, which are injected by the CLI at runtime (dynamically). That's why we are unable to see these files in our “index.html”.  
   
  
Whenever ng-serve builds our application, it creates “bundles” and automatically adds these to our index.html file at runtime. So, from these bundles, the first code has to be executed from “main.ts” file, i.e., “main.ts” file is the main file from where the execution of an Angular application will start.

Have a look at the code this file (main.ts).

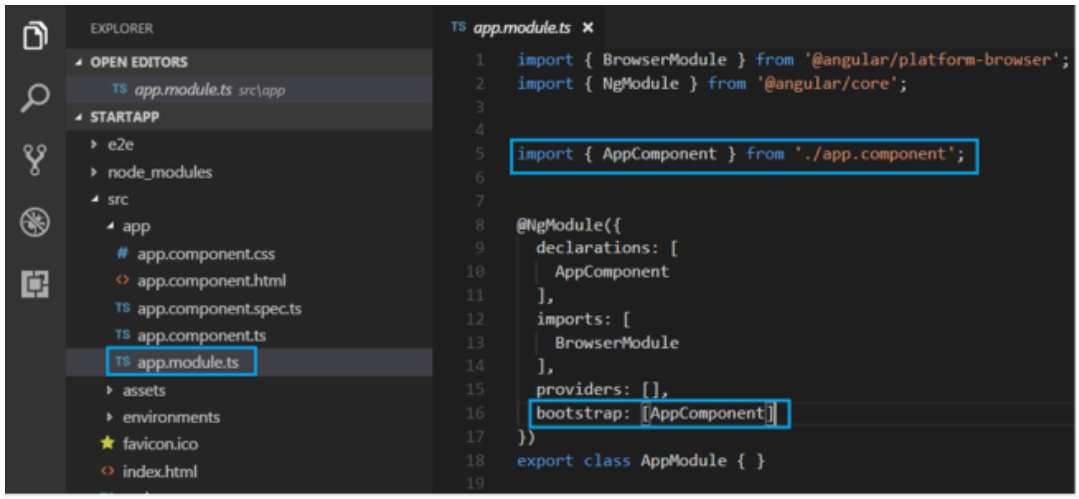
1. **import** { enableProdMode } from '@angular/core';
2. **import** { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
4. **import** { AppModule } from './app/app.module';
5. **import** { environment } from './environments/environment';
7. **if** (environment.production) {
8. enableProdMode();
9. }
11. platformBrowserDynamic().bootstrapModule(AppModule)
12. .**catch**(err => console.log(err));

 See these two points on the below picture in the same file.  
  
  
  
In this file, this “bootstrap” method is important which bootstraps (start) our Angular application.

This refers to AppModule, which looks into the app folders. If we look into “app.module” file, here, we can see a bootstrap array which is basically a list of all the components which should be known to the compiler at the point of time it analyzes our index.html file.

1. **import** { BrowserModule } from '@angular/platform-browser';
2. **import** { NgModule } from '@angular/core';
4. **import** { AppComponent } from './app.component';
6. @NgModule({
7. declarations: [
8. AppComponent
9. ],
10. imports: [
11. BrowserModule
12. ],
13. providers: [],
14. bootstrap: [AppComponent]
15. })
17. **export** **class** AppModule { }

 Let's understand this concept in the below picture.



From the above blog, let us just quickly revise how Angular application gets loaded.

