# Angular 8 Introduction

Angular is the most popular JavaScript framework and platform for developing client-side (front-end) mobile and desktop web apps or single page applications (SPAs).

Angular community has released its latest version known as Angular 8. If you are familiar with previous version of Angular, it will not be difficult for you. You can easily upgrade your older version of Angular to latest version Angular 8.

## What is Angular 8?

Angular 8 is an open-source, client-side TypeScript based JavaScript framework. It is written in TypeScript and complied into JavaScript. Angular 8 is used to create **dynamic web applications**. It is very similar to its previous versions except having some extensive features.

### What is a dynamic web application?

A dynamic web application is simply a dynamic website. i.e. [www.gmail.com](https://www.gmail.com/), [www.facebook.com](https://www.facebook.com/), [www.yahoo.com](https://www.yahoo.com/) etc. which has a tendency to change data/information with respect to 3 parameters:

Play Video

* Time-to-time (eg. news update webs applications)
* Location-to-location (eg. Weather-report web applications)
* User-to-user (eg. Gmail, Facebook type applications)

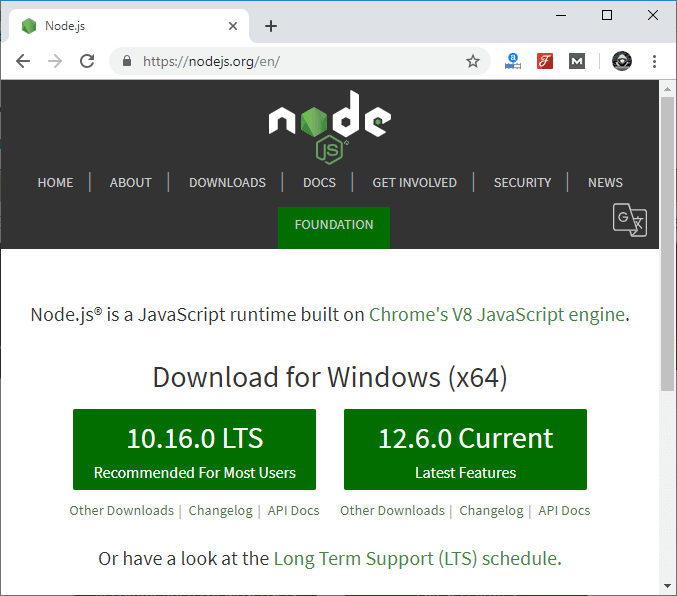
# Angular 8 Installation

(How to install Angular 8 or Angular 8 Environment setup)

Before to setup environment for Angular development using the Angular CLI tool, you must have installed Node.js on your system and set a development environment and npm package manager.

## Install Node.js

Angular requires Node.js version 10.9.0 or later. You can download it from <https://nodejs.org/en/>



After downloading, you have to install it on your system.

Play Videox[](https://campaign.adpushup.com/get-started/?utm_source=banner&utm_campaign=growth_hack)

Step1)

[https://nodejs.org/download/release/v14.2.0/](https://st1.zoom.us/web_client/9zdhk1t/html/externalLinkPage.html?ref=https://nodejs.org/download/release/v14.2.0/)

download .msi with 64 bit

2) in cmd type node –-version

3) npm –v

4) npm install -g @angular/cli@^8.0.0

5) ng version

Visual studio:

[https://code.visualstudio.com/](https://st1.zoom.us/web_client/9zdhk1t/html/externalLinkPage.html?ref=https://code.visualstudio.com/)

# Hello World Angular 8

### Create an app:

**Command:**

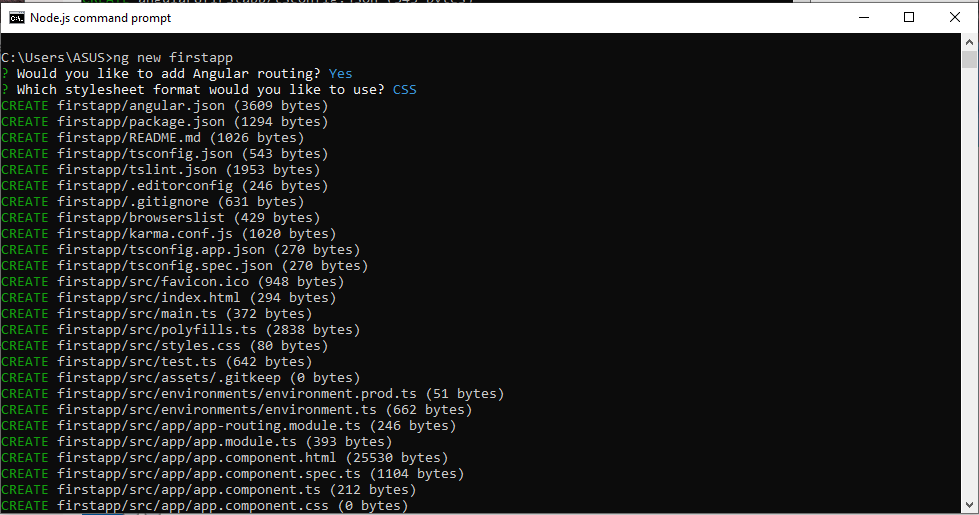
ng new app\_name

**Example:**

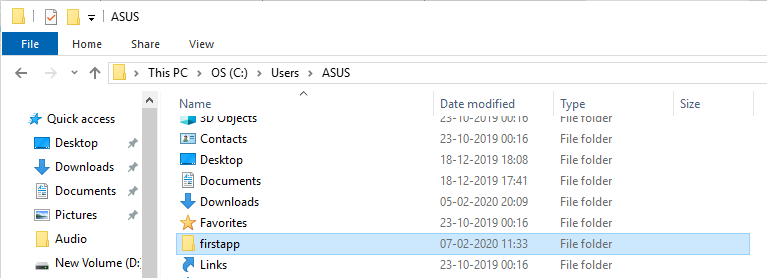
* Create an app and name it as “firstapp”.

**Command:**

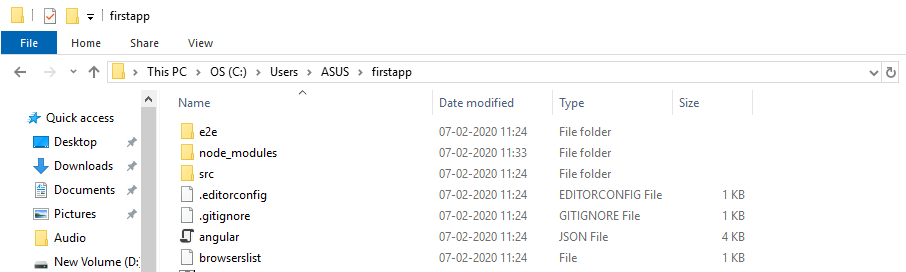
ng new firstapp



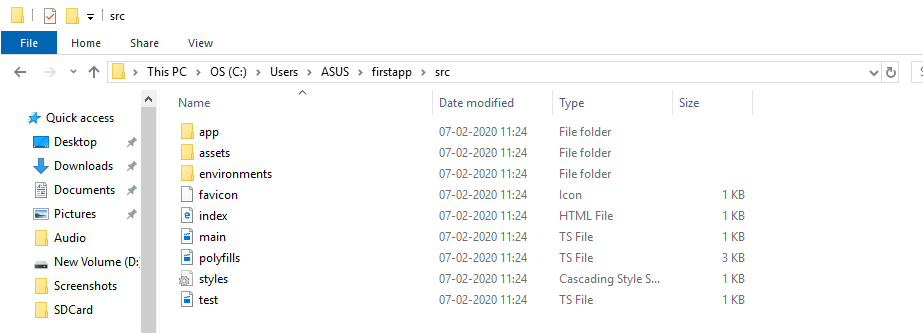
* A folder is thus created.



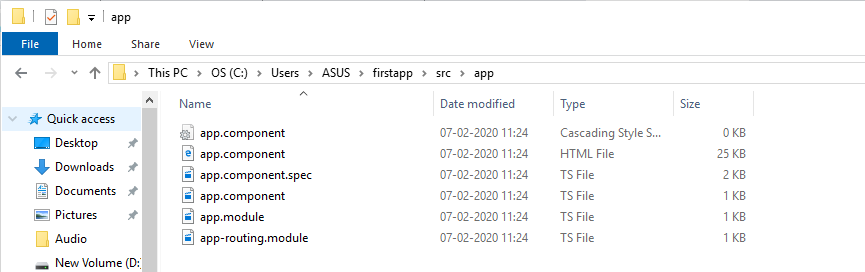
* Open the folder to check subfolders.



* The src is the main folder of a project.
* Open the src folder to check its subfolders.



* The app folder is the root of the app.
* Open the app folder to check its subfolders that include some .ts, HTML and CSS files.



## Files used in Angular 8 App folder:

Angular 8 App folders and files that are mainly used in a project are:

* **src folder:**It includes the main code files related to an angular application.
* **app folder:**It includes the files that are created for the app components.
* **app.component.css:**It includes the cascading style sheets code for the app component.
* **app.component.html:**It includes the HTML file related to an app component, which is a template file used for the data binding by angular.
* **app.component.spec.ts:**Being a unit testing file related to the app component, it is used along with other unit tests. This file is run by the command ng test from Angular CLI.
* **app.component.ts:**Being the most important typescript file, it contains the view logic behind the component.
* **app.module.ts:**As a typescript file, it contains all the dependencies for the website. To determine the needed modules to be imported, the components to be declared and the main component to be bootstrapped, this file is utilized.

## Install Visual Studio Code IDE or JetBrains WebStorm:

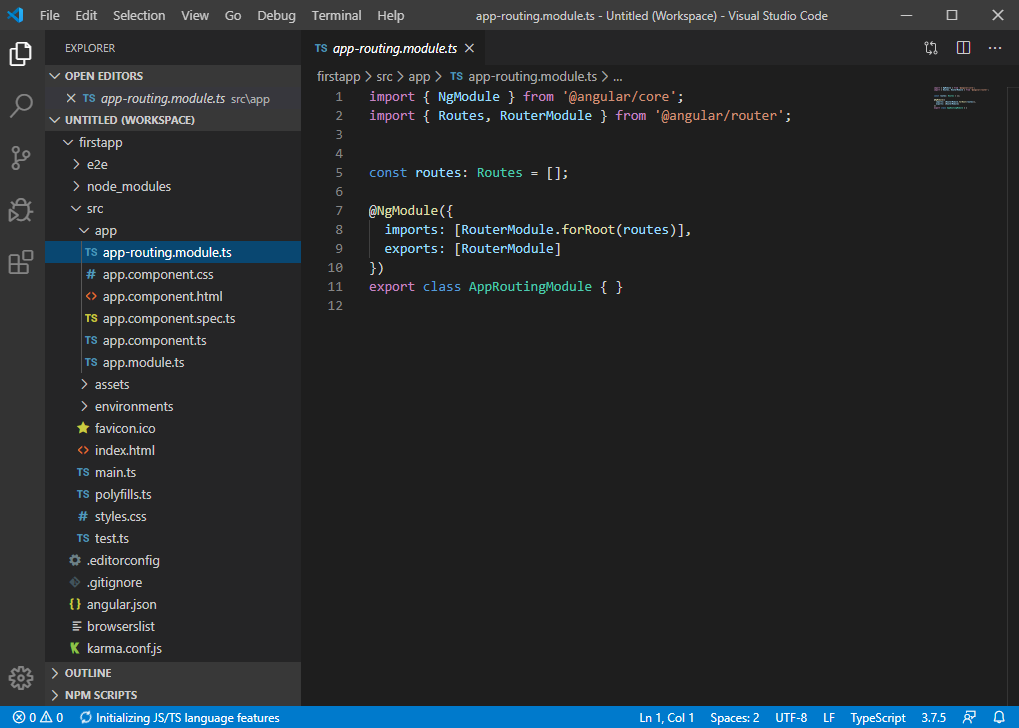
Visual Studio Code IDE or JetBrains WebStorm is the IDE required to run an Angular 8 app.

## Visual Studio Code IDE:

Being light and easy to set up, and with a great range of built-in code editing, formatting, and refactoring features, Visual Studio Code is a popular IDE used for Angular for the development of the app. Along with being free to use, a huge number of extensions are also facilitated which certainly increases productivity.

To download the VS Code go to the below link:

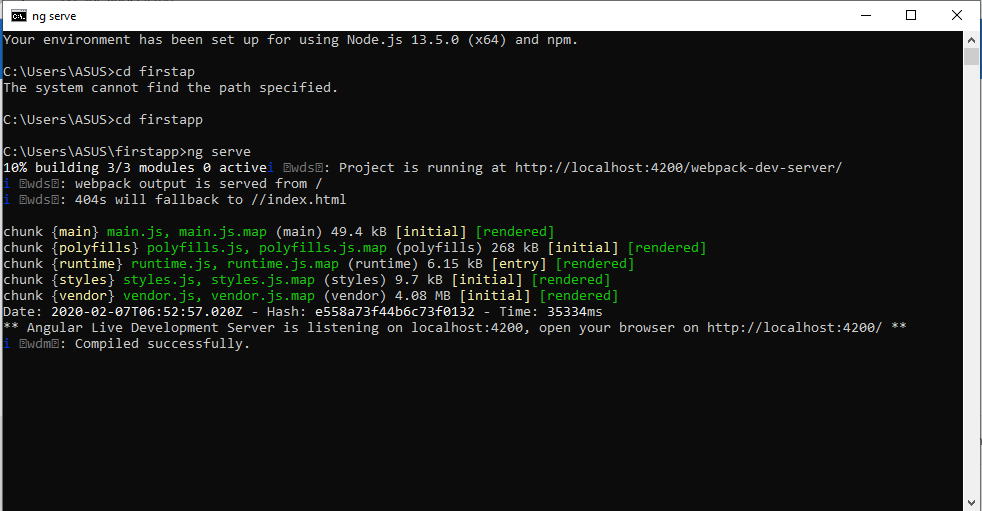
[https://code.visualstudio.com](https://code.visualstudio.com/).



## Run the app:

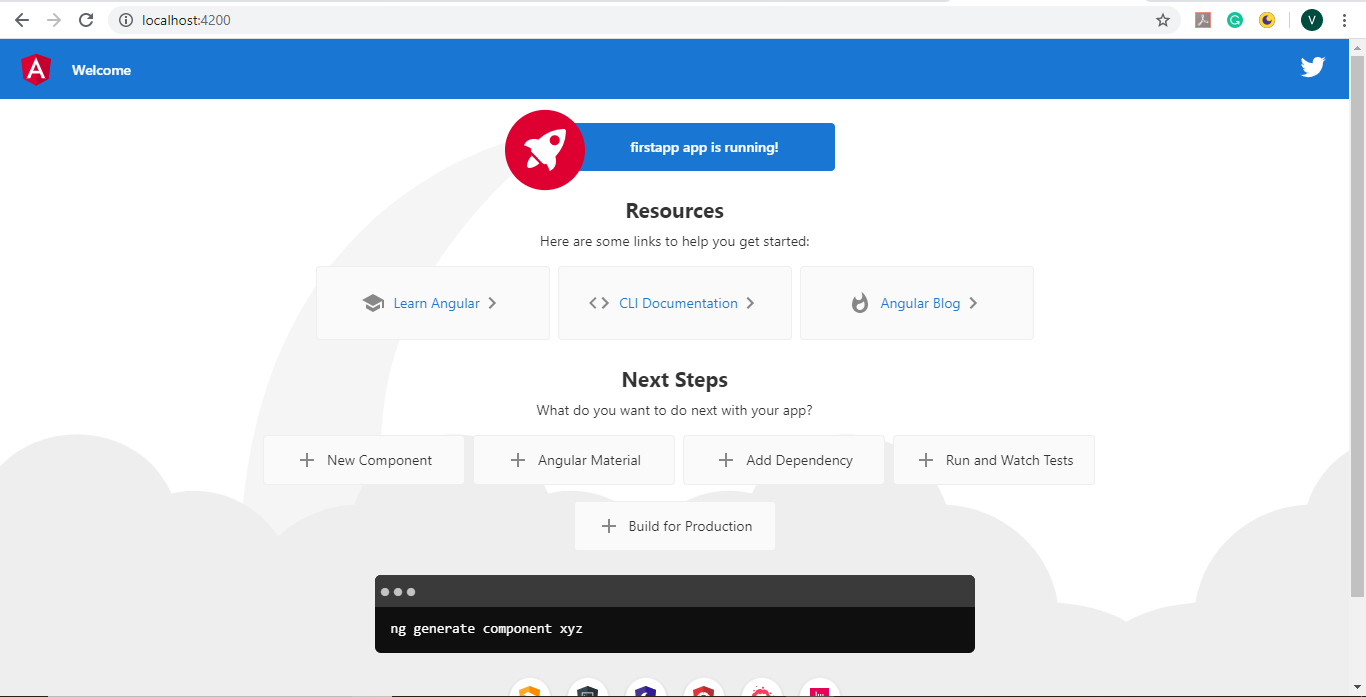
* Open the node.js command prompt.
* Go to the project using the cd command.
* Compile and run the app using the below command.

ng serve



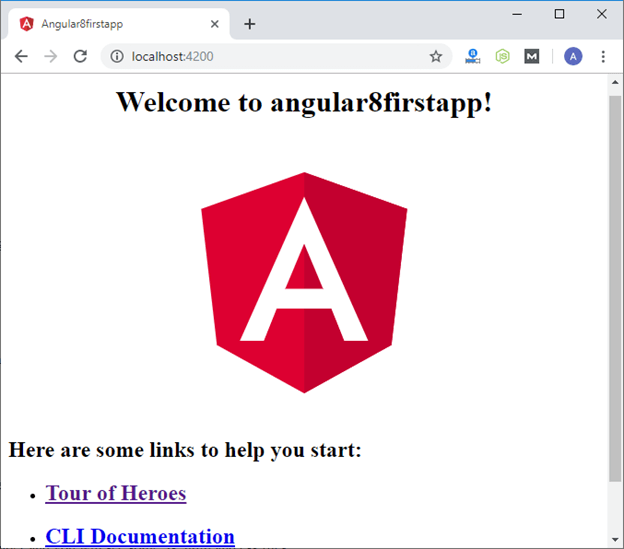
* Open the browser.
* Go to the localhost: <http://localhost:4200/>

The app is thus running now.



# How an Angular's app get loaded and started

When you create an Angular app and run it by using ng serve command, it looks like the following image.



It is a simple Angular app created by using ng new app\_name command and nothing is edited in the app. The name of the app is angular8firstapp.

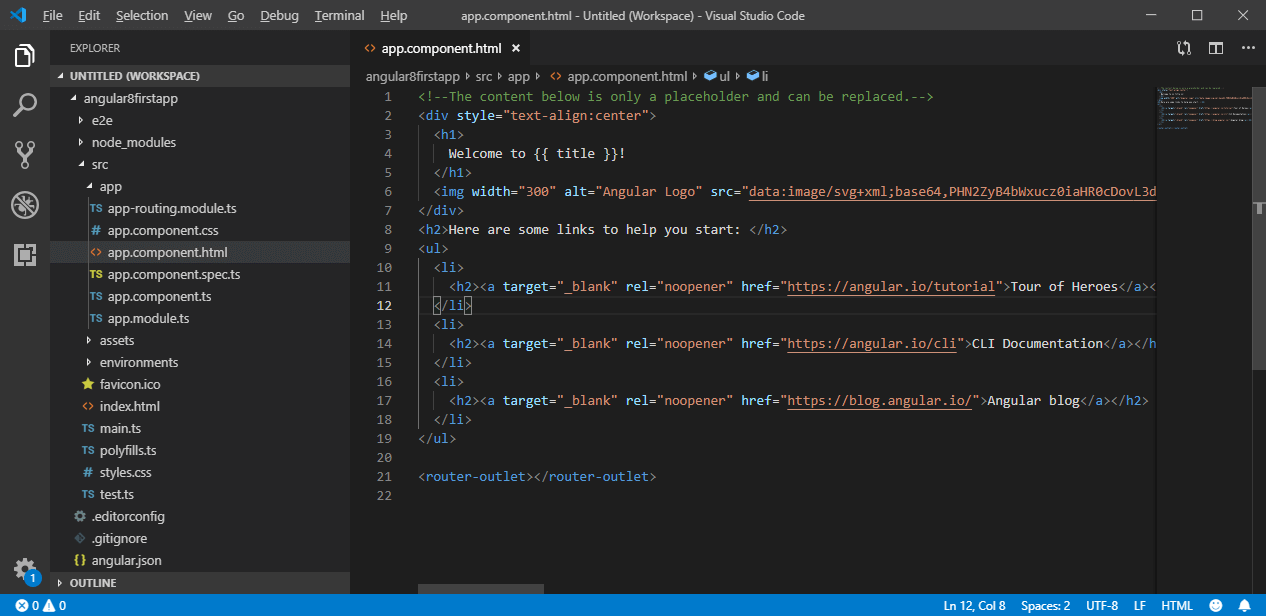
Now, we will learn how the Angular's app is loaded and started.

Let's remove all the code from the app.compoment.html file and write some basic HTML code. For example:

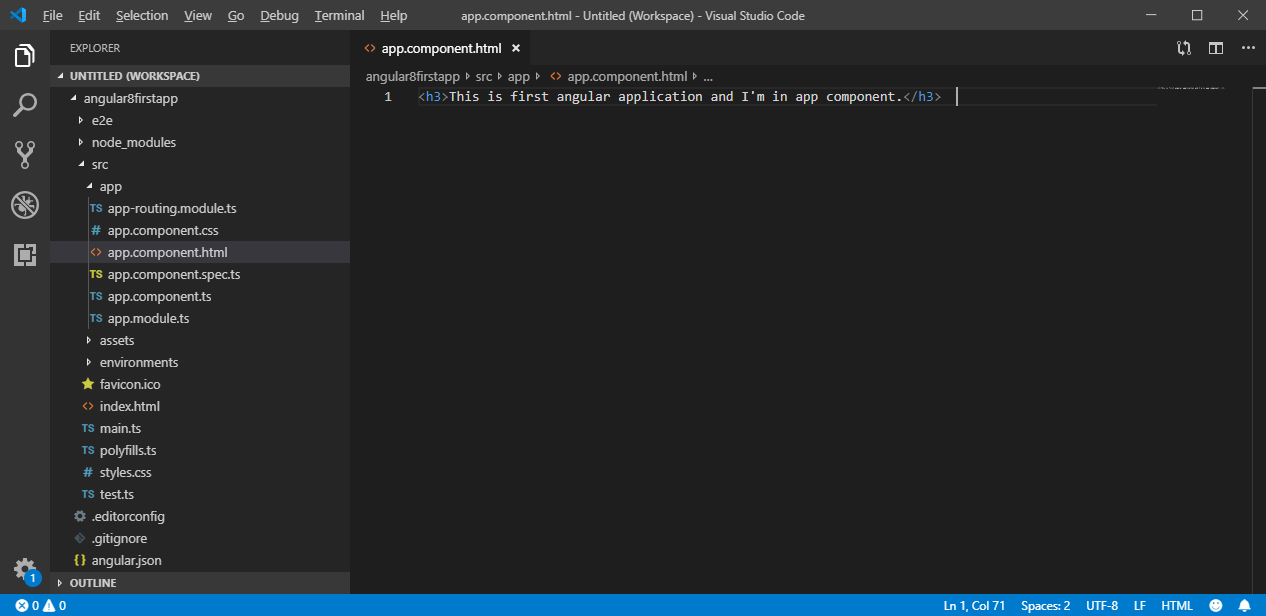
Play Videox[](https://campaign.adpushup.com/get-started/?utm_source=banner&utm_campaign=growth_hack)

1. **<h3>**This is first angular application and I'm in app component.**</h3>**

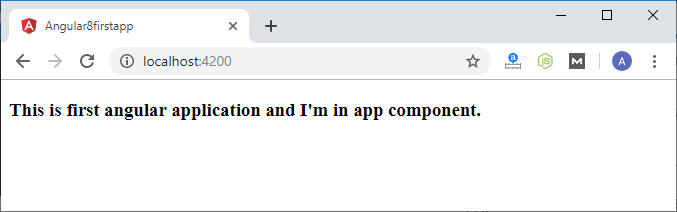
This is the original code in the app.compoment.html file



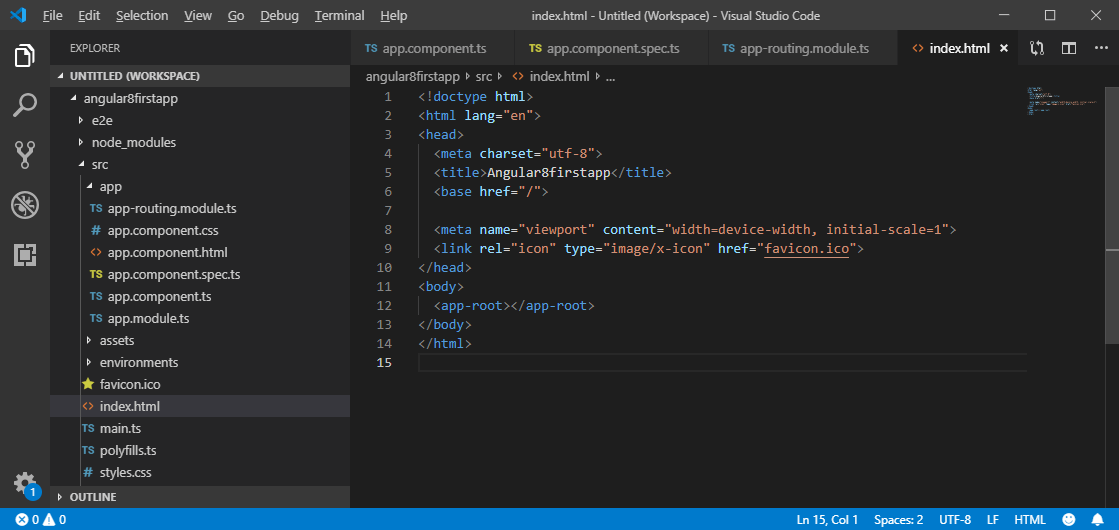
Now, it is replaced and looked like this:



You can also see it in browser:



Here, the above file is not served by the server. The server served an **index.html** file.



Angular is a framework which allows us to create "Single Page Applications", and here the **index.html** is the single page which was provided by the server.

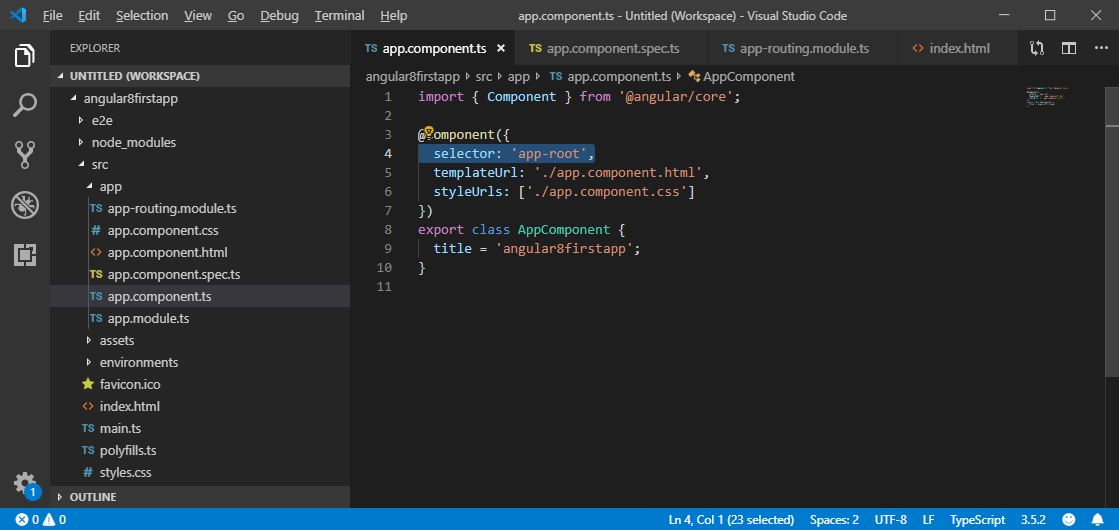
**Index.html:**

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <meta charset="utf-8">
5. <title>Angular8firstapp</title>
6. <base href="/">
8. <meta name="viewport" content="width=device-width, initial-scale=1">
9. <link rel="icon" type="image/x-icon" href="favicon.ico">
10. </head>
11. <body>
12. <app-root></app-root>
13. </body>
14. </html>

The above code looks like a normal HTML code and here the <title> tag shows the same title in the browser as the app's title. But the <body> code is different from normal HTML code. Here, you see "<app-root>" tag 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, see the "**app.component.ts**" file. It is a TypeScript file. Here, you see the "selector" property.

1. **import** { Component } from '@angular/core';
2. @Component({
3. selector: 'app-root',
4. templateUrl: './app.component.html',
5. styleUrls: ['./app.component.css']
6. })
7. export **class** AppComponent {
8. title = 'angular8firstapp';
9. }



You can see that the selector property contains the string as index.html file. This information is required 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.

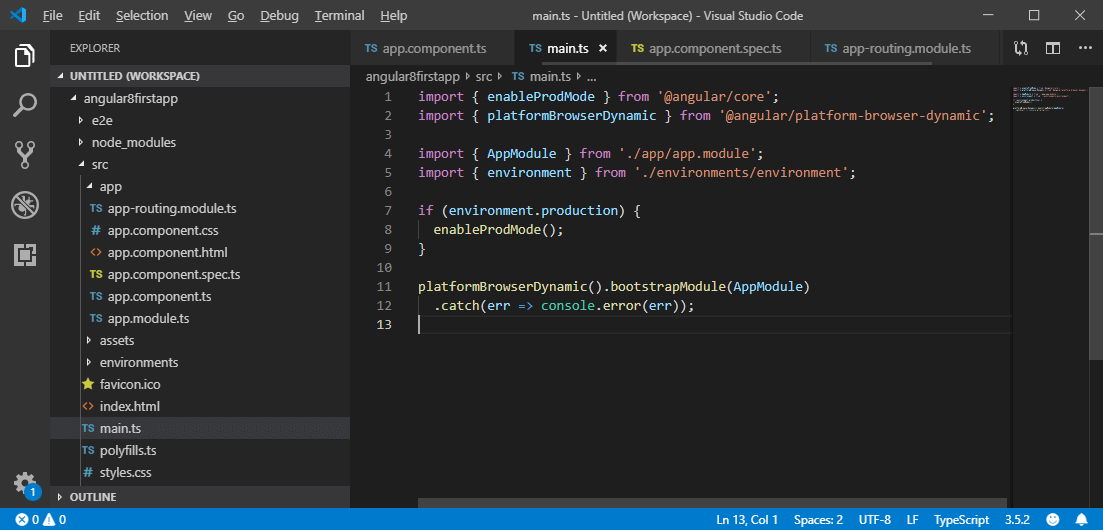
Now, you see how an "app-root" is included in index.html file. Now, let's see "How does Angular trigger?"

Whenever ng-serve builds the application, it creates "bundles" and automatically adds these to 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 starts.

main.ts >> app.Module.ts >> app.component.ts >> index.html >> app.component.html

**Main.ts file:**

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

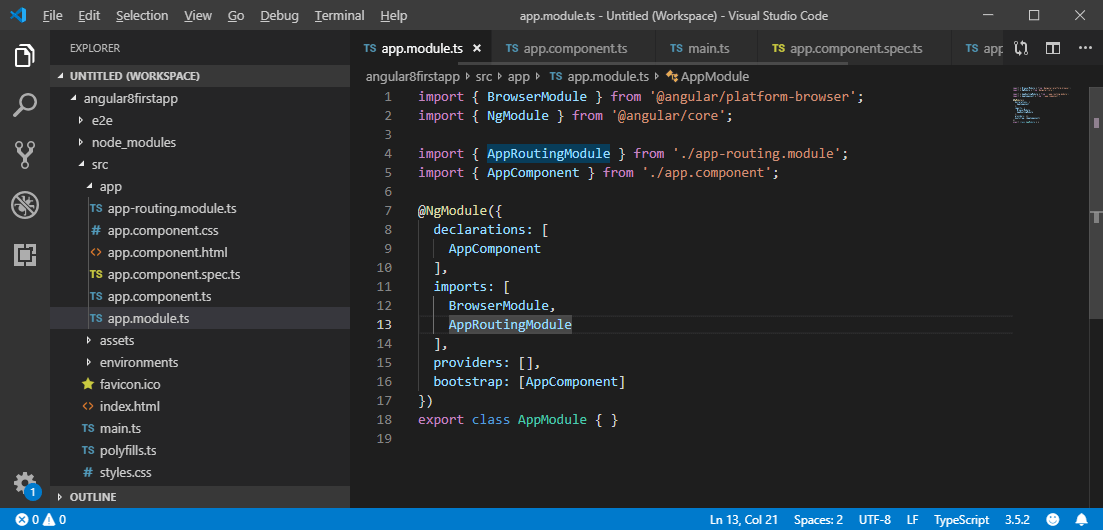


**Explanation:**

In the above angular code, the bootstrap method starts the Angular application which refers to the AppModule. It looks into the app folders.

**See app.module.ts file:**

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



**Explanation:**

In the above file, a bootstrap array analyzes the index.html file. Here, the bootstrap array is a list of all the components.

Thus, an Angular application gets loaded as:

main.ts >> app.Module.ts >> app.component.ts >> index.html >> app.component.html

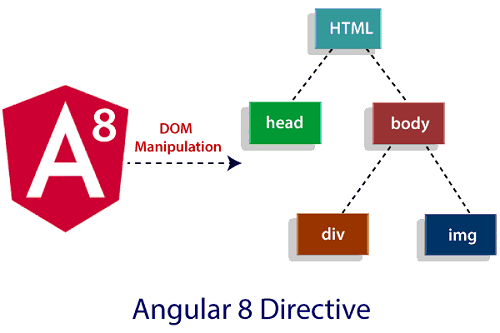
Now, you can see that the Angular application gets loaded as:

1. main.ts  **>>**   app.Module.ts  **>>**  app.component.ts  **>>**  index.html  **>>**  app.component.html

in visual studio u have to use npm install and npm start in terminal after importing the project

# Angular 8 Directives

The Angular 8 directives are used to manipulate the DOM. By using Angular directives, you can change the appearance, behavior or a layout of a DOM element. It also helps you to extend HTML.



**Angular 8 directives can be classified in 3 categories based on how they behave:**

* Component Directives
* Structural Directives
* Attribute Directives

**Component Directives:** Component directives are used in main class. They contain the detail of how the component should be processed, instantiated and used at runtime.

**Structural Directives:** Structural directives start with a \* sign. These directives are used to manipulate and change the structure of the DOM elements. For example, \*ngIf directive, \*ngSwitch directive, and \*ngFor directive.

Play Videox

* **\*ngIf Directive:** The ngIf allows us to Add/Remove DOM Element.
* **\*ngSwitch Directive:** The \*ngSwitch allows us to Add/Remove DOM Element. It is similar to switch statement of C#.
* **\*ngFor Directive:** The \*ngFor directive is used to repeat a portion of HTML template once per each item from an iterable list (Collection).

**Attribute Directives:** Attribute directives are used to change the look and behavior of the DOM elements. For example: ngClass directive, and ngStyle directive etc.

* **ngClass Directive:** The ngClass directive is used to add or remove CSS classes to an HTML element.
* **ngStyle Directive:** The ngStyle directive facilitates you to modify the style of an HTML element using the expression. You can also use ngStyle directive to dynamically change the style of your HTML element.

# Angular 8 ngIf Directive

The ngIf Directives is used to add or remove HTML Elements according to the expression. The expression must return a Boolean value. If the expression is false then the element is removed, otherwise element is inserted. It is similar to the ng-if directive of AngularJS.

### ngIf Syntax

1. <p \*ngIf="condition">
2. condition is **true** and ngIf is **true**.
3. </p>
4. <p \*ngIf="!condition">
5. condition is **false** and ngIf is **false**.
6. </p>

### The \*ngIf directive form with an "else" block

1. <div \*ngIf="condition; else elseBlock">
2. Content to render when condition is **true**.
3. </div>
4. <ng-template #elseBlock>
5. Content to render when condition is **false**.
6. </ng-template>

The ngIf directive does not hide the DOM element. It removes the entire element along with its subtree from the DOM. It also removes the corresponding state freeing up the resources attached to the element.

The \*ngIf directive is most commonly used to conditionally show an inline template. See the following example:

1. @Component({
2. selector: 'ng-if-simple',
3. template: `
4. <button (click)="show = !show">{{show ? 'hide' : 'show'}}</button>
5. show = {{show}}
6. <br>
7. <div \*ngIf="show">Text to show</div>
8. `
9. })
10. export **class** NgIfSimple {
11. show: **boolean** = **true**;
12. }

### Same template example with else block

1. @Component({
2. selector: 'ng-if-else',
3. template: `
4. <button (click)="show = !show">{{show ? 'hide' : 'show'}}</button>
5. show = {{show}}
6. <br>
7. <div \*ngIf="show; else elseBlock">Text to show</div>
8. <ng-template #elseBlock>Alternate text **while** primary text is hidden</ng-template>
9. `
10. })
11. export **class** NgIfElse {
12. show: **boolean** = **true**;
13. }

# Angular 8 \*ngFor Directive

The \*ngFor directive is used to repeat a portion of HTML template once per each item from an iterable list (Collection). The ngFor is an Angular structural directive and is similar to ngRepeat in AngularJS. Some local variables like Index, First, Last, odd and even are exported by \*ngFor directive.

## Syntax of ngFor

See the simplified syntax for the ngFor directive:

1. <li \*ngFor="let item of items;"> .... </li>

## How to use ngFor Directive?

To Use ngFor directive, you have to create a block of HTML elements, which can display a single item of the items collection. After that you can use the ngFor directive to tell angular to repeat that block of HTML elements for each item in the list.

### Example for \*ngFor Directive

First, you have to create an angular Application. After that open the app.component.ts and add the following code.

Play Videox[](https://campaign.adpushup.com/get-started/?utm_source=banner&utm_campaign=growth_hack)

The following Code contains a list of Top 3 movies in a movies array. Let's build a template to display these movies in a tabular form.

1. **import** { Component } from '@angular/core';
2. @Component({
3. selector: 'movie-app',
4. templateUrl:'./app/app.component.html',
5. styleUrls:['./app/app.component.css']
6. })
7. export **class** AppComponent
8. {
9. title: string ="Top 10 Movies" ;
10. movies: Movie[] =[
11. {title:'Zootopia',director:'Byron Howard, Rich Moore',cast:'Idris Elba, Ginnifer Goodwin, Jason Bateman',releaseDate:'March 4, 2016'},
12. {title:'Batman v Superman: Dawn of Justice',director:'Zack Snyder',cast:'Ben Affleck, Henry Cavill, Amy Adams',releaseDate:'March 25, 2016'},
13. {title:'Captain America: Civil War',director:'Anthony Russo, Joe Russo',cast:'Scarlett Johansson, Elizabeth Olsen, Chris Evans',releaseDate:'May 6, 2016'},
14. {title:'X-Men: Apocalypse',director:'Bryan Singer',cast:'Jennifer Lawrence, Olivia Munn, Oscar Isaac',releaseDate:'May 27, 2016'},
15. ]
16. }
17. **class** Movie {
18. title : string;
19. director : string;
20. cast : string;
21. releaseDate : string;
22. }

Now, open the app. component.html and add the following code:

1. <div **class**='panel panel-primary'>
2. <div **class**='panel-heading'>
3. {{title}}
4. </div>
5. <div **class**='panel-body'>
6. <div **class**='table-responsive'>
7. <table **class**='table'>
8. <thead>
9. <tr>
10. <th>Title</th>
11. <th>Director</th>
12. <th>Cast</th>
13. <th>Release Date</th>
14. </tr>
15. </thead>
16. <tbody>
17. <tr \*ngFor="let movie of movies;">
18. <td>{{movie.title}}</td>
19. <td>{{movie.director}}</td>
20. <td>{{movie.cast}}</td>
21. <td>{{movie.releaseDate}}</td>
22. </tr>
23. </tbody>
24. </table>
25. </div>
26. </div>
27. </div>

When you run the application, It will show the movies in tabular form.

# Data Binding in Angular 8

Data binding is the core concept of Angular 8 and used to define the communication between a component and the DOM.

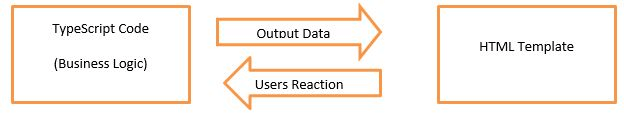
## One-way databinding

One way databinding is a simple one way communication where HTML template is changed when we make changes in TypeScript code.

## Two-way databinding

In two-way databinding, automatic synchronization of data happens between the Model and the View. Here, change is reflected in both components. Whenever you make changes in the Model, it will be reflected in the View and when you make changes in View, it will be reflected in Model.

This happens immediately and automatically, ensures that the HTML template and the TypeScript code are updated at all times.



# Property Binding in Angular 8

Property Binding is also a **one-way data binding** technique. In property binding, we bind a property of a DOM element to a field which is a defined property in our component TypeScript code. Actually Angular internally converts string interpolation into property binding.

**For example:**

<img [src]="imgUrl" />

Property binding is preferred over string interpolation because it has shorter and cleaner code String interpolation should be used when you want to simply display some dynamic data from a component on the view between headings like h1, h2, p etc.

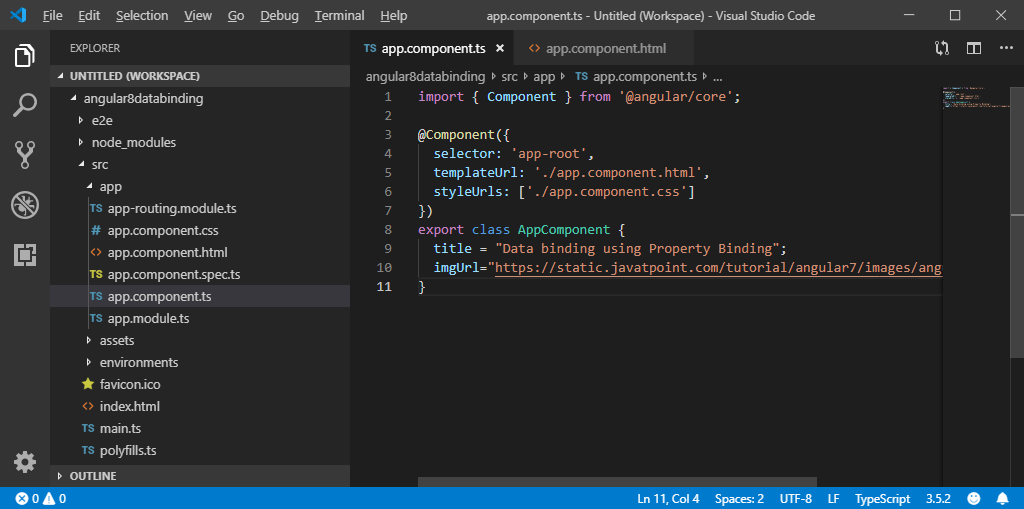
Play Videox[](https://campaign.adpushup.com/get-started/?utm_source=banner&utm_campaign=growth_hack)

#### Note: String Interpolation and Property binding both are one-way binding. Means, if field value in the component changes, Angular will automatically update the DOM. But any changes in the DOM will not be reflected back in the component.

## Property Binding Example

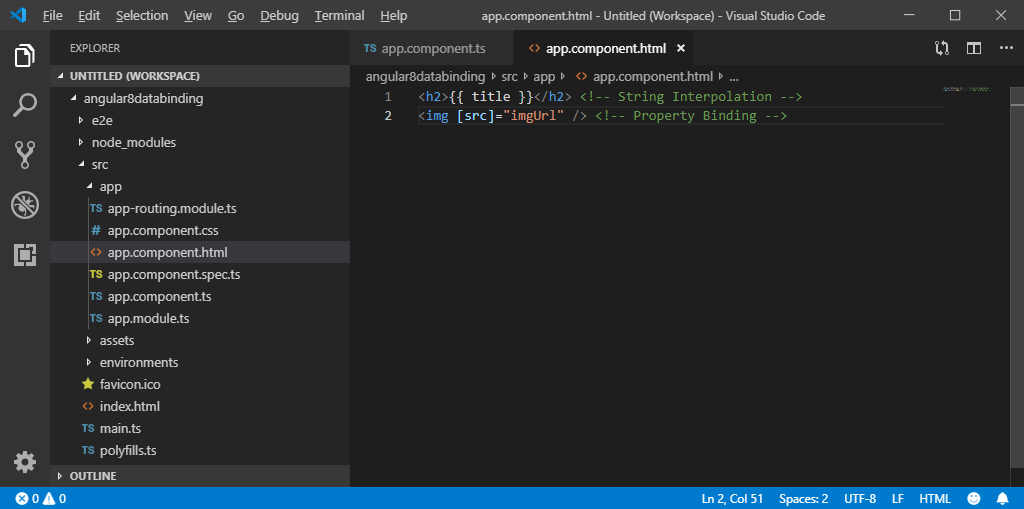
Open **app.componnt.ts** file and add the following code:

1. import { Component } from '@angular/core';
2. @Component({
3. selector: 'app-root',
4. templateUrl: './app.component.html',
5. styleUrls: ['./app.component.css']
6. })
7. export class AppComponent {
8. title = "Data binding using Property Binding";
9. imgUrl="https://static.javatpoint.com/tutorial/angular7/images/angular-7-logo.png";
10. }



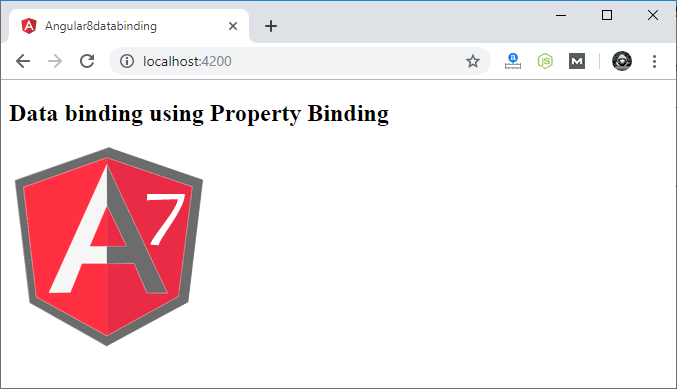
Now, open **app.component.html** and use the following code for property binding:

1. **<h2>**{{ title }}**</h2>** <!-- String Interpolation -->
2. **<img** [src]="imgUrl" **/>** <!-- Property Binding -->



Run the ng serve command and open local host to see the result.

**Output:**



# Event Binding in Angular 8

In Angular 8, event binding is used to handle the events raised from the DOM like button click, mouse move etc. When the DOM event happens (eg. click, change, keyup), it calls the specified method in the component. In the following example, the cookBacon() method from the component is called when the button is clicked:

**For example:**

1. **<button** (click)="cookBacon()"**></button>**

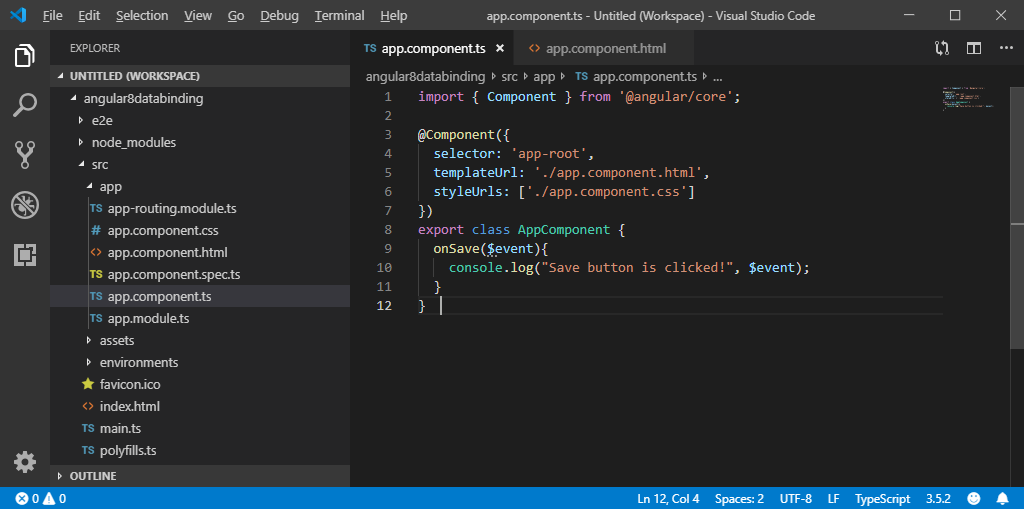
### Event Binding Example

Let's take a button in the HTML template and handle the click event of this button. To implement event binding, we will bind click event of a button with a method of the component.

Now, open the **app.component.ts** file and use the following code:

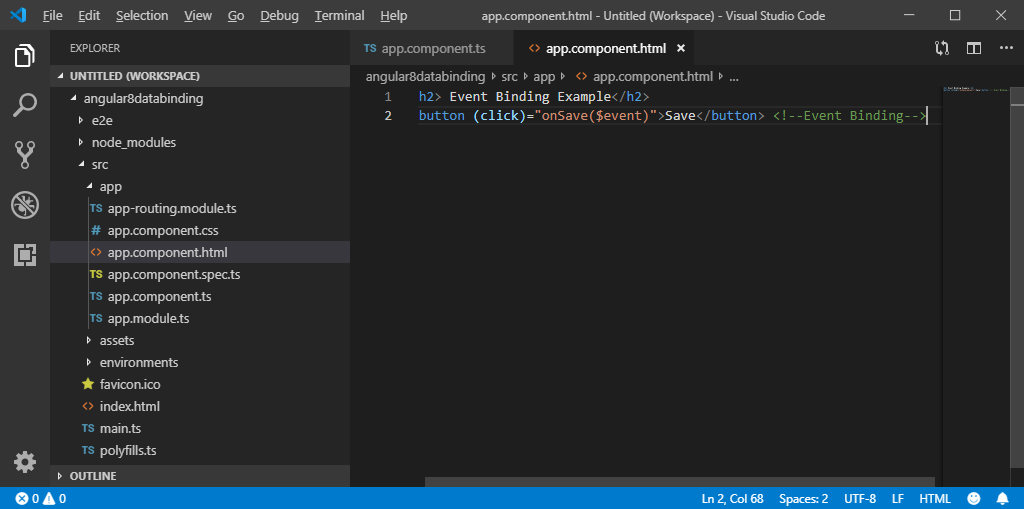
Play Videox[](https://campaign.adpushup.com/get-started/?utm_source=banner&utm_campaign=growth_hack)

1. import { Component } from '@angular/core';
2. @Component({
3. selector: 'app-root',
4. templateUrl: './app.component.html',
5. styleUrls: ['./app.component.css']
6. })
7. export class AppComponent {
8. onSave($event){
9. console.log("Save button is clicked!", $event);
10. }
11. }

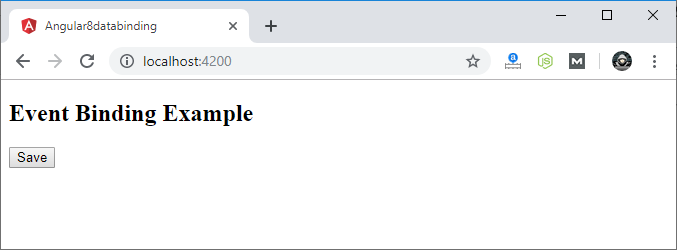


**app.component.html:**

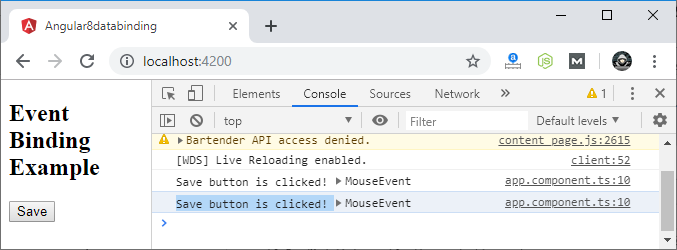
1. **<h2>** Event Binding Example**</h2>**
2. **<button** (click)="onSave($event)"**>**Save**</button>** <!--Event Binding-->



**Output:**



Click on the "Save" button and open console to see result.



Now, you can see that the "Save" button is clicked.

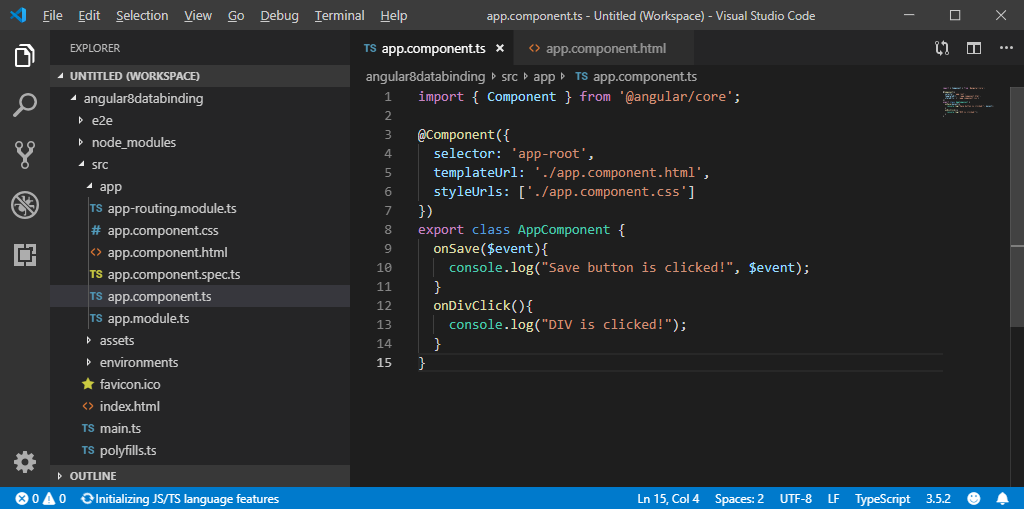
## Event Bubbling

Event bubbling is used to specify an order in which event handlers are called when one element is nested inside a second element, and both elements have registered a listener for the same event (i.e. click).

Let's see the above button example. Here, I have used a div wrapper around the button in component HTML and div has also a click event handler. It is only to show some message if div is clicked.

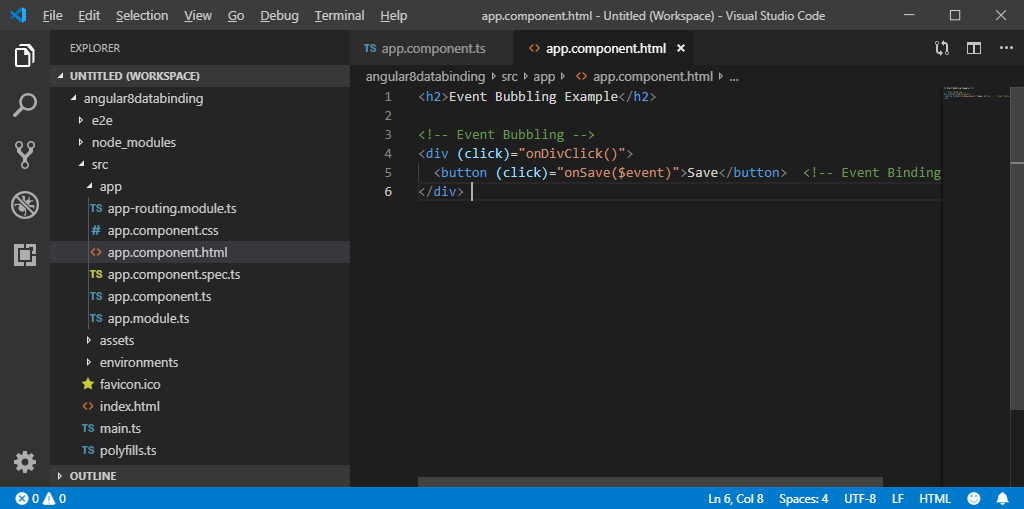
Use the following code in **app.component.ts** file:

1. import { Component } from '@angular/core';
2. @Component({
3. selector: 'app-root',
4. templateUrl: './app.component.html',
5. styleUrls: ['./app.component.css']
6. })
7. export class AppComponent {
8. onSave($event){
9. console.log("Save button is clicked!", $event);
10. }
11. onDivClick(){
12. console.log("DIV is clicked!");
13. }
14. }



**app.component.html:**

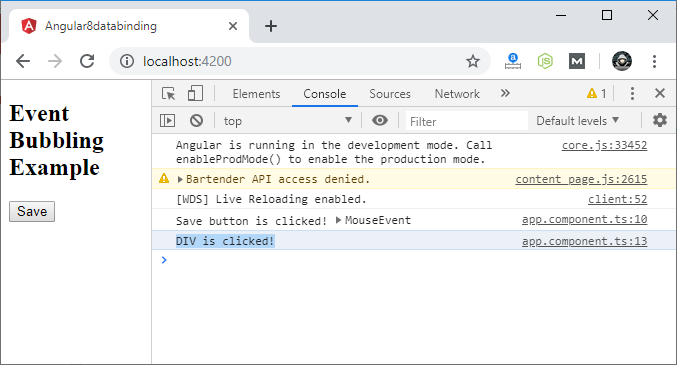
1. **<h2>**Event Bubbling Example**</h2>**
2. <!-- Event Bubbling -->
3. **<div** (click)="onDivClick()"**>**
4. **<button** (click)="onSave($event)"**>**Save**</button>**  <!-- Event Binding -->
5. **</div>**



**Output:**



Click on the "Save" button and open console to see result.



Here, you can see that your div message is also occurred. This is all due to event bubbling where you have specified onDivClick button.

# Two way Data Binding in Angular 8

We have seen that in one-way data binding any change in the template (view) were not be reflected in the component TypeScript code. To resolve this problem, Angular provides two-way data binding. The two-way binding has a feature to update data from component to view and vice-versa.

In two-way databinding, automatic synchronization of data happens between the Model and the View. Here, change is reflected in both components. Whenever you make changes in the Model, it will be reflected in the View and when you make changes in View, it will be reflected in Model.

This happens immediately and automatically, ensures that the HTML template and the TypeScript code are updated at all times.

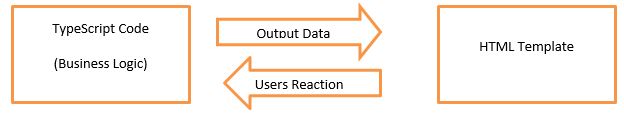
In two way data binding, **property binding and event binding** are combined together.

Play Videox[](https://campaign.adpushup.com/get-started/?utm_source=banner&utm_campaign=growth_hack)

### Syntax:

1. [(ngModel)] = "[property of your component]"

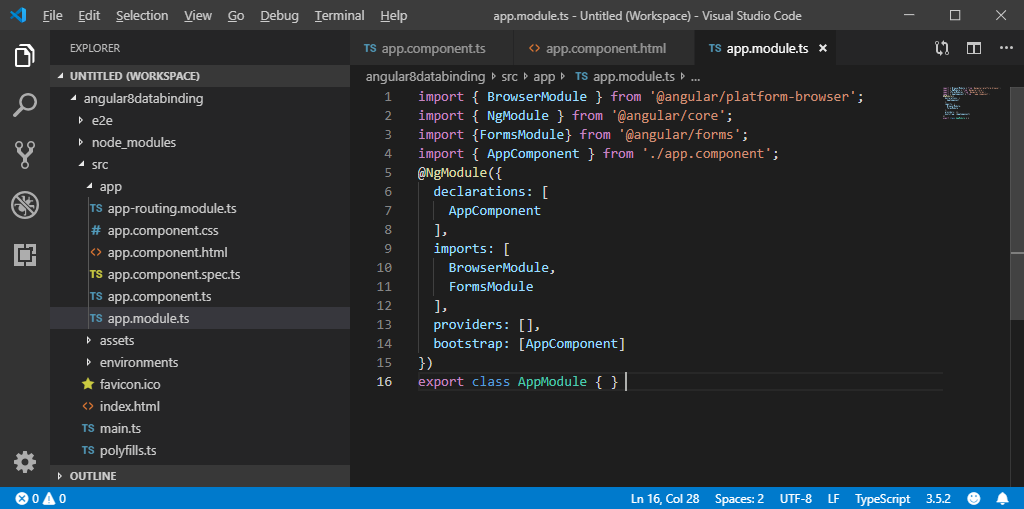
#### Note: For two way data binding, we have to enable the ngModel directive. It depends upon FormsModule in angular/forms package, so we have to add FormsModule in imports[] array in the AppModule.



Let's take an example to understand it better.

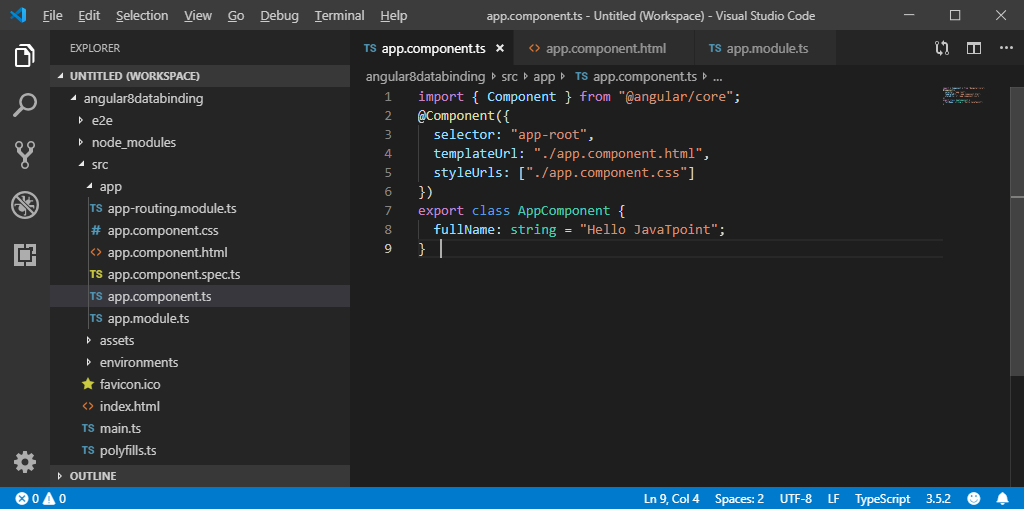
Open your project's **app.module.ts** file and use the following code:

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



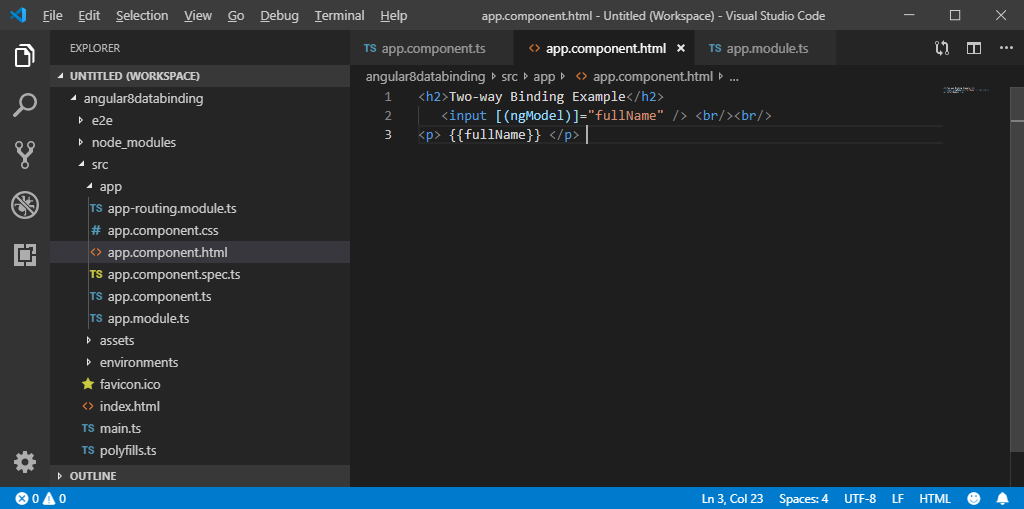
**app.component.ts file:**

1. import { Component } from "@angular/core";
2. @Component({
3. selector: "app-root",
4. templateUrl: "./app.component.html",
5. styleUrls: ["./app.component.css"]
6. })
7. export class AppComponent {
8. fullName: string = "Hello JavaTpoint";
9. }



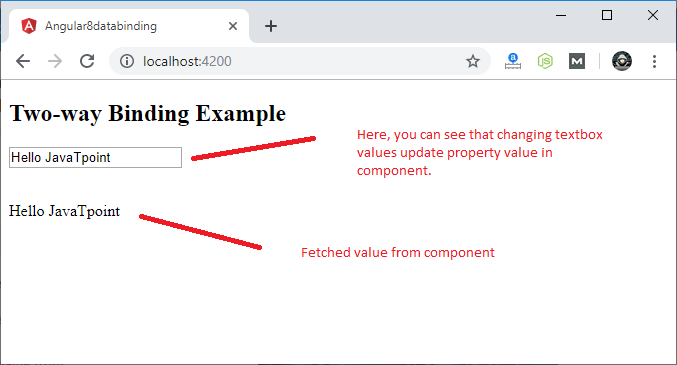
**app.component.html file:**

1. **<h2>**Two-way Binding Example**</h2>**
2. **<input** [(ngModel)]="fullName" **/>** **<br/><br/>**
3. **<p>** {{fullName}} **</p>**



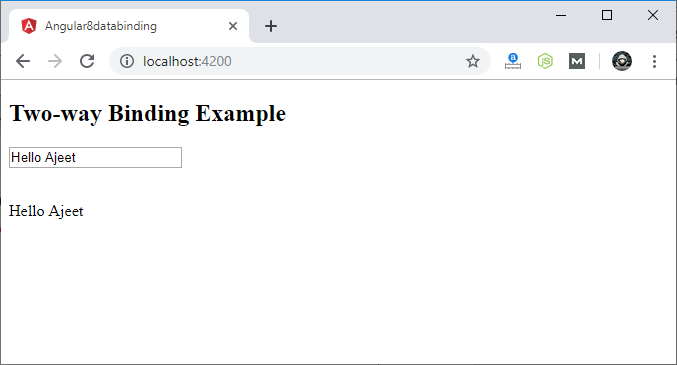
Now, start your server and open local host browser to see the result.

**Output:**



You can check it by changing textbox value and it will be updated in component as well.

**For example:**



# Angular 8 Forms

Angular forms are used to handle user's input. We can use Angular form in our application to enable users to log in, to update profile, to enter information, and to perform many other data-entry tasks.

In Angular 8, there are 2 approaches to handle user's input through forms:

* Reactive forms
* Template-driven forms

Both approaches are used to collect user input events from the view, validate the user input, create a form model and data model to update, and provide a way to track changes.

## Reactive Forms vs. Template-driven Forms

Both Reactive forms and Template-driven forms manage and process data differently. Each offers different advantages.

Play Videox[](https://campaign.adpushup.com/get-started/?utm_source=banner&utm_campaign=growth_hack)

### Reactive Forms

* Reactive forms are more robust.
* Reactive forms are more scalable, reusable, and testable.
* They are most preferred to use if forms are a key part of your application, or your application is already built using reactive patterns. In both cases, reactive forms are best to use.

### Template-driven Forms

* Template-driven forms are best if you want to add a simple form to your application. **For example:** email list signup form.
* Template-driven forms are easy to use in the application but they are not as scalable as Reactive forms.
* Template-driven forms are mainly used if your application's requires a very basic form and logic. It can easily be managed in a template.

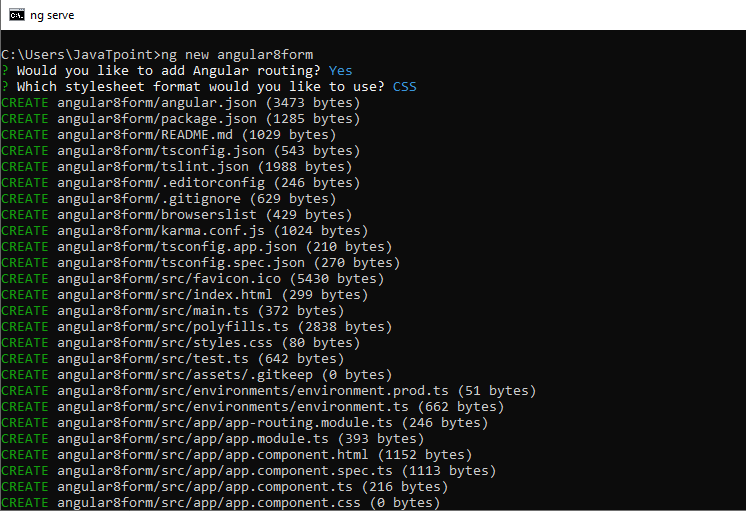
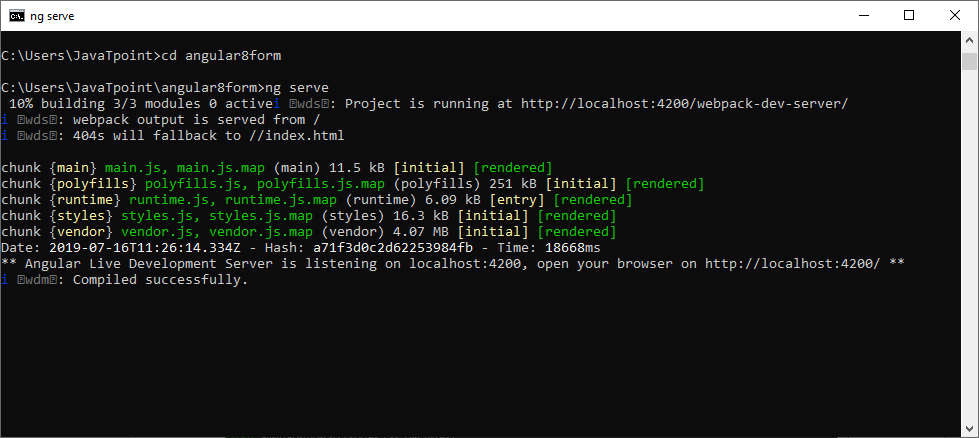
## Angular 8 Form Example

Let's understand the Angular 8 form by creating a form example. Here, we use Angular reactive form.

**Follow the steps given below:**

* Create an Angular form app named **angular8from** and run the server by using the following commands.

1. ng new angular8form
2. cd angular8form
3. ng serve

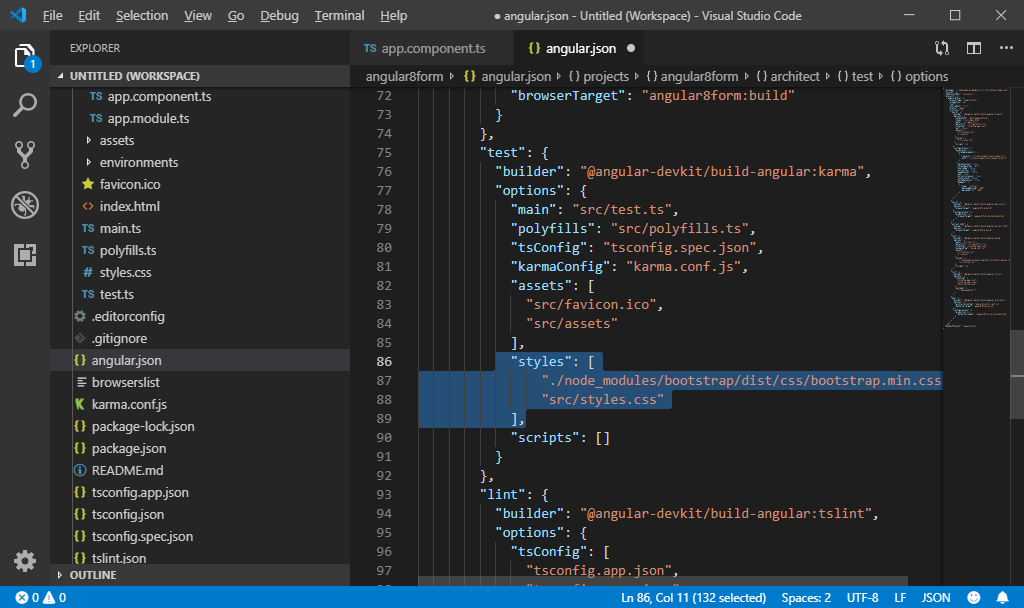
  


* Install the Bootstrap 4 using the following command.

1. npm install bootstrap --save

Now, include the bootstrap 4 inside the angular.json file inside styles array.

1. "styles": [
2. "./node\_modules/bootstrap/dist/css/bootstrap.min.css",
3. "src/styles.css"
4. ],

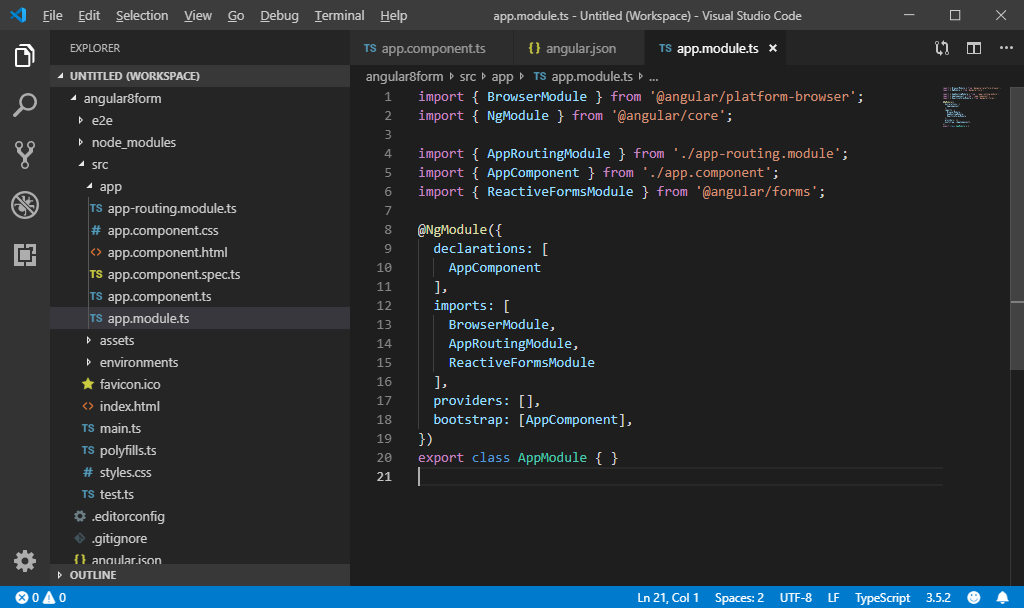


* Register the Reactive Forms Module

Use the reactive forms by importing ReactiveFormsModule from the @angular/forms package and add it to your app.module.ts file's imports array.

So use the following code inside the app.module.ts file.

1. // app.module.ts
2. import { BrowserModule } from '@angular/platform-browser';
3. import { NgModule } from '@angular/core';
5. import { AppRoutingModule } from './app-routing.module';
6. import { AppComponent } from './app.component';
7. import { ReactiveFormsModule } from '@angular/forms';
9. @NgModule({
10. declarations: [
11. AppComponent
12. ],
13. imports: [
14. BrowserModule,
15. AppRoutingModule,
16. ReactiveFormsModule
17. ],
18. providers: [],
19. bootstrap: [AppComponent],
20. })
21. export class AppModule { }

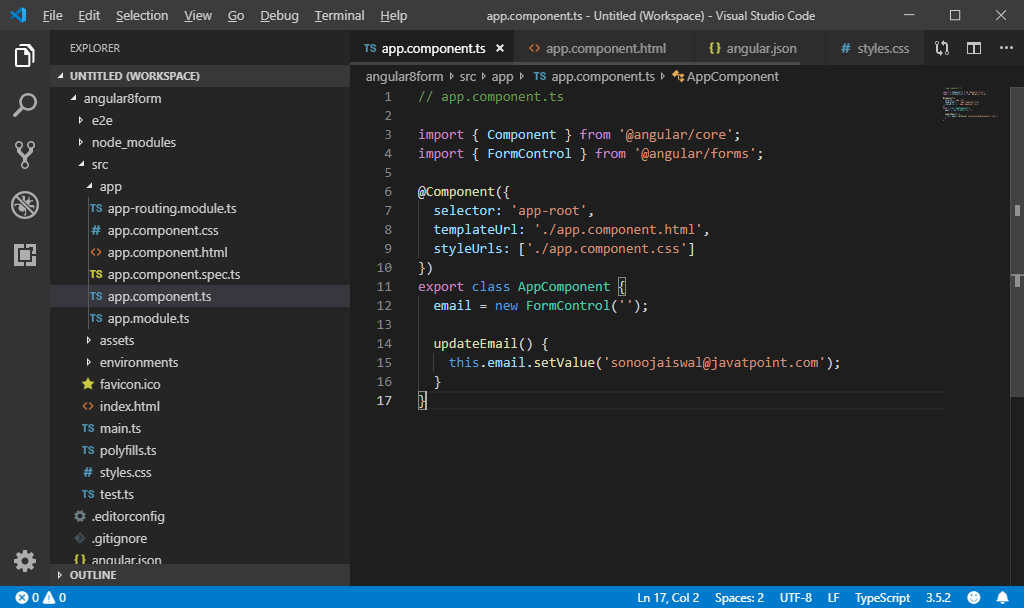


* Add FormControl class register the control into the template and update the FormControl value

The FormControl class is the fundamental building block when using the reactive forms. So if you want to register the single form control, you need to import the FormControl class into your component and create the new instance of a form control to save as the class property.

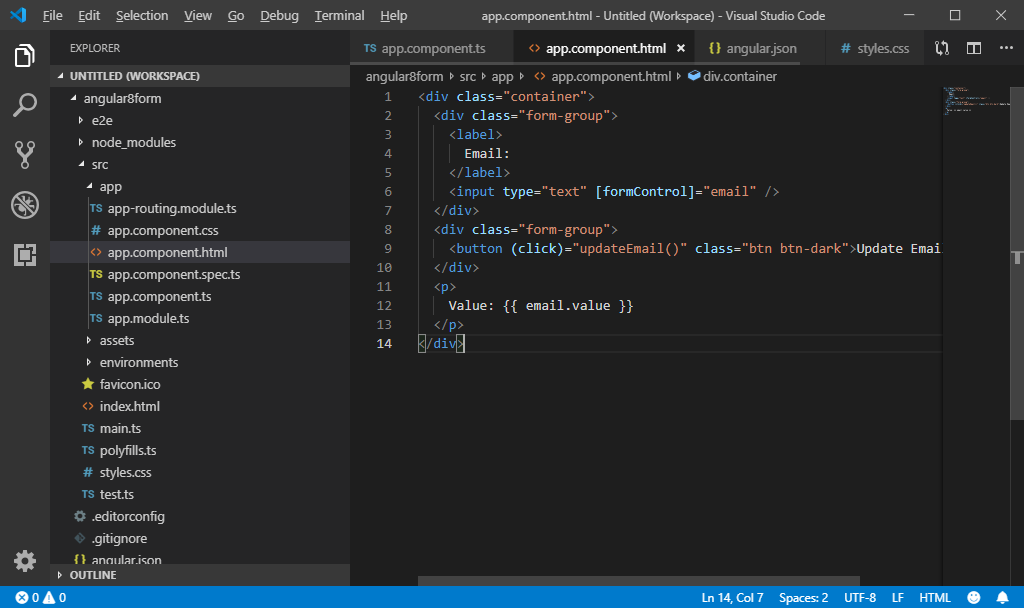
Now, modify the **app.component.ts** file.

1. // app.component.ts
2. import { Component } from '@angular/core';
3. import { FormControl } from '@angular/forms';
4. @Component({
5. selector: 'app-root',
6. templateUrl: './app.component.html',
7. styleUrls: ['./app.component.css']
8. })
9. export class AppComponent {
10. email = new FormControl('');
11. updateEmail() {
12. this.email.setValue('sonoojaiswal@javatpoint.com');
13. }
14. }



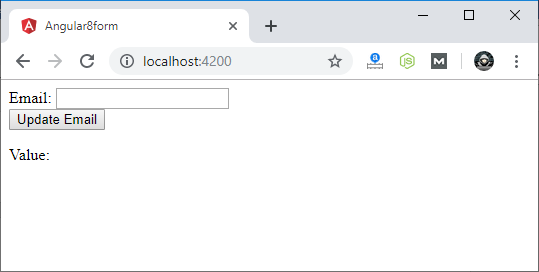
Also, update the view **app.component.html** file.

1. **<div** class="container"**>**
2. **<div** class="form-group"**>**
3. **<label>**
4. Email:
5. **</label>**
6. **<input** type="text" [formControl]="email" **/>**
7. **</div>**
8. **<div** class="form-group"**>**
9. **<button** (click)="updateEmail()" class="btn btn-dark"**>**Update Email**</button>**
10. **</div>**
11. **<p>**
12. Value: {{ email.value }}
13. **</p>**
14. **</div>**

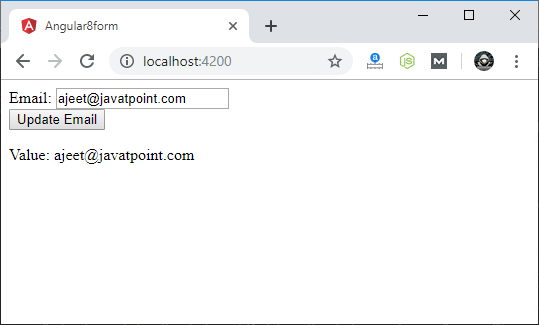


Now, save your code and start the server.

**Output:**



Enter any email id and you will see the result in the value.



When you click on the "Update Email" button, it will update the email id as we saved in the template file.

