Angular 8

## What is Angular 8?

Angular 8 is a client-side TypeScript based framework which is used to create dynamic web applications. It is very similar to its previous versions except having some extensive features.

**These are the most prominent features of Angular 8:**

* Angular 8 supports TypeScript 3.4
* Angular 8 supports Web Workers
* The new compiler for Angular 8 is Ivy Rendering Engine
* Angular 8 provides dynamic imports for lazy-loaded modules.
* Improvement of ngUpgrade

# 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:

Hello Java Program for Beginners

* 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 Previous Versions

* **AngularJS (also known as Angular 1.0):** AngularJS is a JavaScript based open-source frontend web framework developed and maintained by Google. AngularJS can be added to an HTML page with a <script> tag. Because AngularJS was the first version of the Angular, so it is also known as Angular 1. AngularJS was first released on October 20, 2010 by a team of Google.
* **Angular 2:** Angular 2 was a complete rewrite of AngularJS. It was first released in May 2016 and the final version was released on September 14, 2016.
* **Angular 4:** Angular 4 was the updated version of Angular 2. The Google team skipped the Angular 3 to avoid the confusion due to the misalignment of the router package's version which was already distributed as v3.3.0.
* **Angular 5:** Angular 5 was the improved version of the previous one. It was released on November 1, 2017 and improved the support for progressive web apps.
* **Angular 6:** Angular 6 Angular 6 was released on May 4, 2018. It was a major released focused on ng update, ng add, Angular Elements, Angular Material + CDK Components, Angular Material Starter Components, CLI Workspaces, Library Support, Tree Shakable Providers, Animations Performance Improvements, and RxJS v6.
* **Angular 7:** Angular 7 was released on October 18, 2018. It was focused on Application Performance, Angular Material & CDK, Virtual Scrolling, Improved Accessibility etc.
* **Angular 8:** Angular 8 is the latest version running nowadays. Angular 8 is released on May 28, 2019. It is mainly focused on Differential loading, Dynamic imports for lazy routes, web workers and Angular Ivy as an opt-in support. It also supports TypeScript 3.4.

# Features and Advantages of Angular 8

The Angular community has released its latest version Angular 8 with an impressive list of changes and improvements including the much awaited Ivy compiler as an opt-in feature.

**Most prominent features of Angular 8:**

* Support TypeScript 3.4
* Supports Web Workers
* Preview of Ivy available
* Lazy loading
* Improvement of ngUpgrade

## TypeScript 3.4

Angular 8 supports TypeScript 3.4 and it is required to run your Angular 8 project. So, you have to upgrade your TypeScript version to 3.4. TypeScript 3.4 introduces a new flag called --incremental. The incremental tells TypeScript to save information about the project graph from the last compilation. Every time the TypeScript is invoked with --incremental, it will use that information to detect the least costly way to type-check and emit changes to your project.

## Web workers class

JavaScript is single threaded, so it is common for more critical tasks like data calls to take place asynchronously. Web Workers facilitates you to run the CPU intensive computations in the background thread, freeing the main thread to update the user interface.

Web workers can also be helpful, if your application is unresponsive while processing data.

If you want to outsource such a calculation to a background, we must first create the web worker using the Angular CLI.

## Preview of Ivy and Bazel available

After the release of Angular 8, a preview version of Ivy is now available for testing. Ivy is the new rendering engine that produces small bundle size and Bazel is the new build system. Both are ready for proper use with Angular 8. The preview of these two should be available shortly. Ivy is a new compiler/runtime of Angular and Angular 8 is a first release to offer a switch to opt-in into Ivy officially.

To use Ivy in your project, you can instruct the Angular CLI to enable Ivy in your project using the --enable-ivy switch:

1. $ ng new angular-project --enable-ivy

Ivy is supposed to be a by default rendering engine in Angular version 9.

Bazel provides one of the newest features of Angular 8 as a possibility to build your CLI application more quickly.

**The main advantages of Bazel are:**

* The incremental build and tests.
* It provides a chance to make your backends and frontends with a same tool.
* It has a possibility to have remote builds and cache on the build farm.
* Dynamic imports for lazy-loaded modules

## Lazy Loading

Angular 8 facilitates you to use standard dynamic import syntax instead of a custom string for lazy-loaded modules.

**It means lazy-loaded import that looked like this:**

1. { path: '/student', loadChildren: './student/student.module#StudentModule' }

**Will be looked like this:**

1. { path: '/student', loadChildren: () =**>** import('./student/student.module').then(s =**>** s.StudentModule) }

## Improvement of ngUpgrade

The Angular CLI is continuously improving. Now, the ng build, ng test and ng run are equipped by 3rd-party libraries and tool. For example, AngularFire already makes use of these new capabilities with a deploy command.

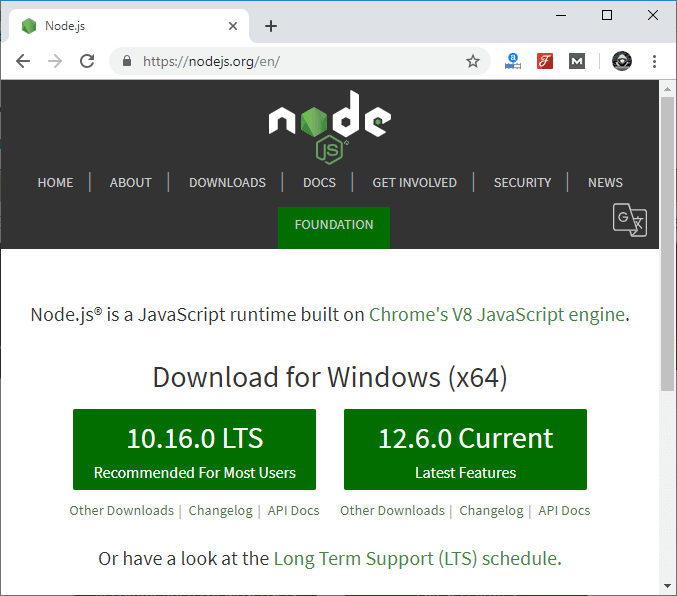
# Angular 8 Installation

(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/>



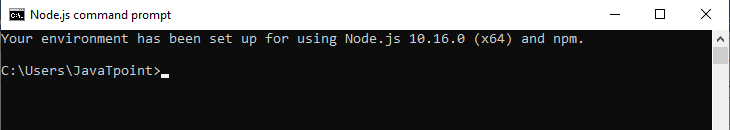
After downloading, you have to install it on your system.

HTML Tutorial

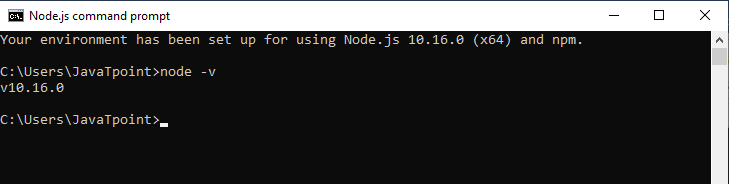
See how to install Node.js on Windows: [**Click Here**](https://www.javatpoint.com/install-nodejs)

See how to install Node.js on Linux/ Ubuntu/ CentOS: [**Click Here**](https://www.javatpoint.com/install-nodejs-on-linux-ubuntu-centos)

Once you have installed Node.js on your system, open node.js command prompt.



* To check your version, run **node -v** in a terminal/console window.



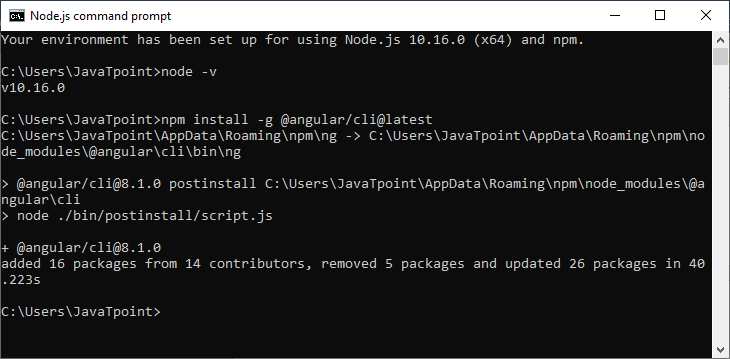
## Use npm to Install Angular CLI

Use the following command to install Angular CLI

1. npm install -g @angular/cli

Or

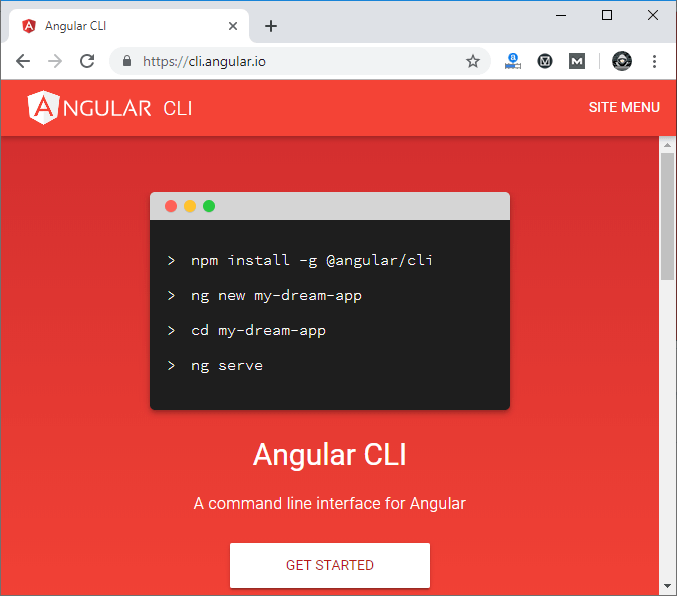
1. npm install -g @angular/cli@latest



Or

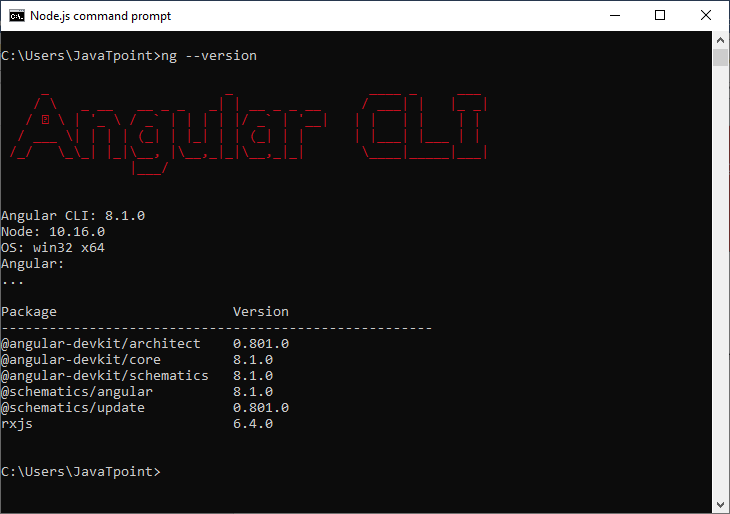
Just go to Angular CLI official website <https://cli.angular.io/>

You will see the whole cli command to create an Angular app. You need to run the first command to install Angular CLI. These steps are same for Windows and Mac.



## Check your Installed versions

* To check Node and Angular CLI version, use ng --version command.



Now, Angular 8 is installed on your system.

# Angular 8 First App

Let's see how to create an Angular 8 application.

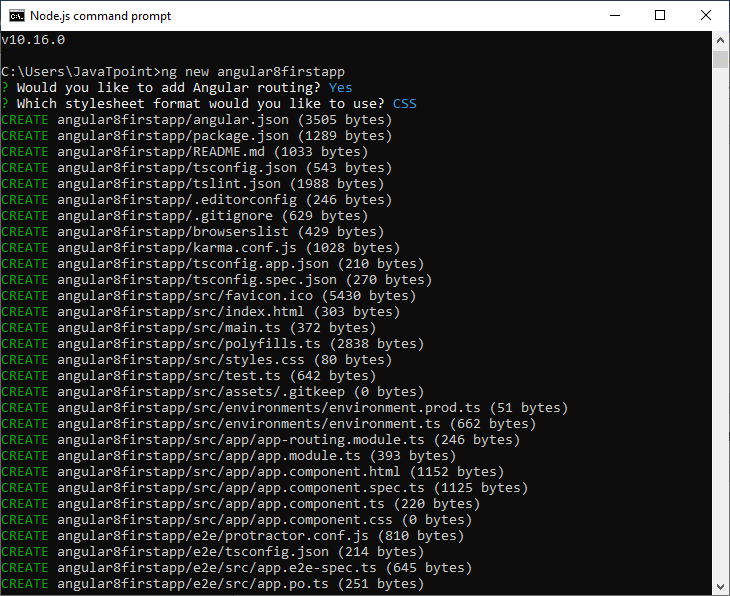
### To create an app

**Syntax:**

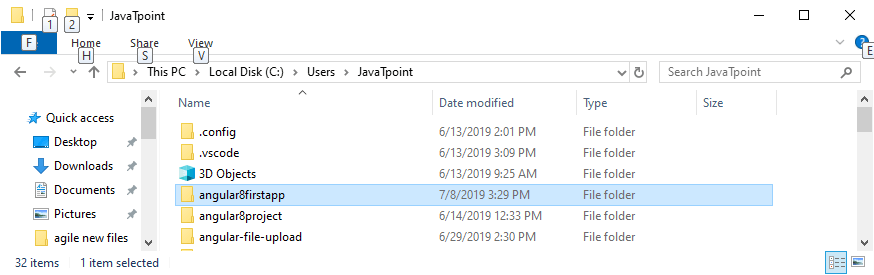
1. ng new app\_name

**For example:** Here, we are going to create an app named "angular8firstapp"

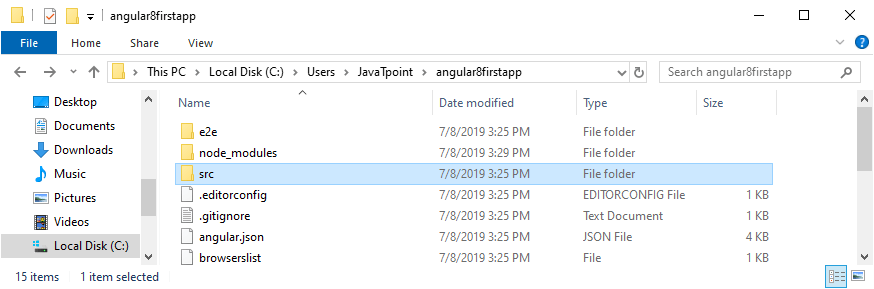
1. ng new angular8firstapp



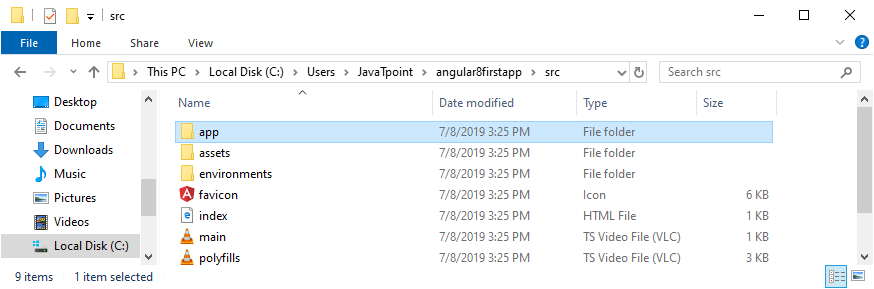
You can see that a folder is created. This is your first created app of Angular 8.



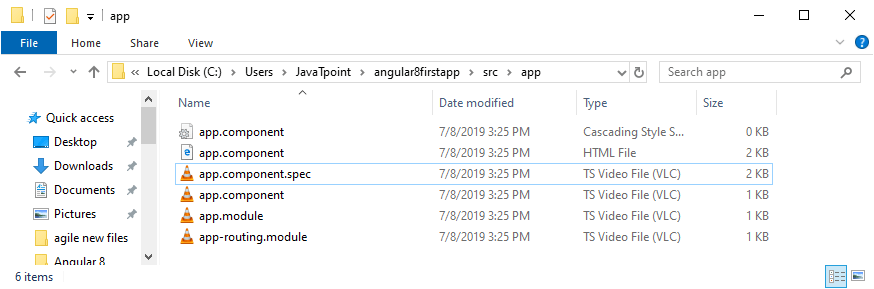
Open this folder and you will see the some subfolders.



Here, **src** is the main folder of your project. Open the src folder and you will see some other subfolders.



This app folder is the root of your Angular 8 app. Open this folder and you will see some .ts, html and css files.



## Files used in Angular 7 App folder

Angular 7 App files which are mainly used in your project are given below:

* **src folder:** This is the folder which contains the main code files related to your angular application.
* **app folder:** The app folder contains the files, you have created for app components.
* **app.component.css:** This file contains the cascading style sheets code for your app component.
* **app.component.html:** This file contains the html file related to app component. This is the template file which is used by angular to do the data binding.
* **app.component.spec.ts:** This file is a unit testing file related to app component. This file is used along with other unit tests. It is run from Angular CLI by the command ng test.
* **app.component.ts:** This is the most important typescript file which includes the view logic behind the component.
* **app.module.ts:** This is also a typescript file which includes all the dependencies for the website. This file is used to define the needed modules to be imported, the components to be declared and the main component to be bootstrapped.

## Install Visual Studio Code IDE or JetBrains WebStorm

You must have an IDE like Visual Studio Code IDE or JetBrains WebStorm to run your Angular 7 app.

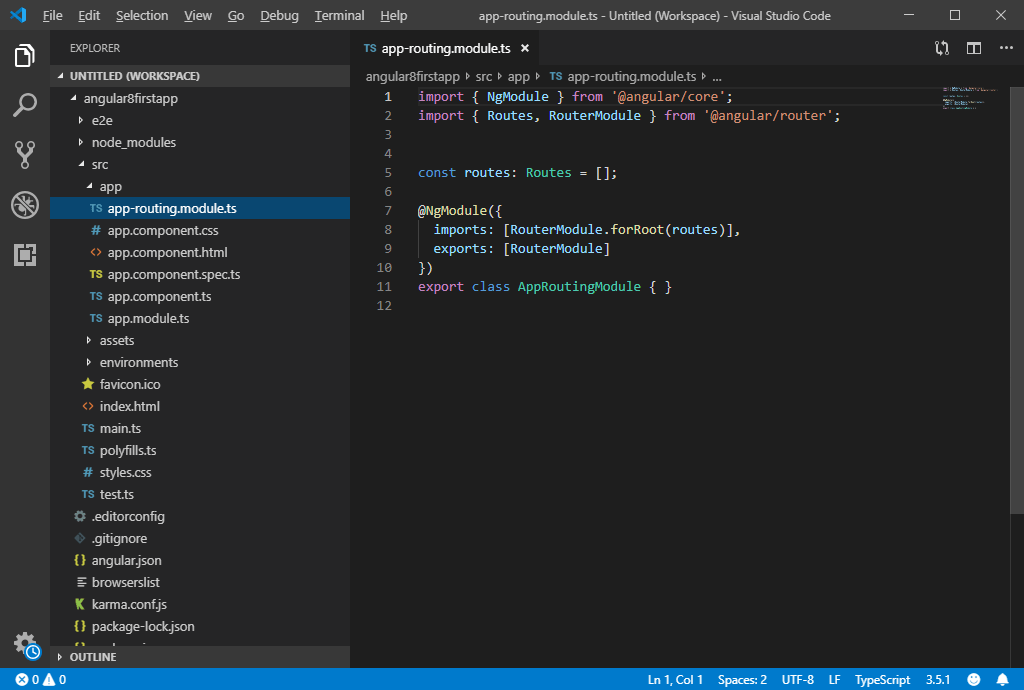
**VS Code** is light and easy to setup, it has a great range of built-in code editing, formatting, and refactoring features. It is free to use. It also provides a huge number of extensions that will significantly increase your productivity.

You can download VS Code from here: [https://code.visualstudio.com](https://code.visualstudio.com/)

**JetBrains WebStorm** is also a great IDE to develop Angular 7 apps. It is fast, attractive, and very easy to use software but, it is not free to use. You have to purchase it later, it only provides a trial period of 30 days for free.

You can download Jetbrains Webstorm from here: [https://www.jetbrains.com/webstorm/download/#section=windows](https://www.jetbrains.com/webstorm/download/#section=windows/)

Here, we are using VS Code IDE:

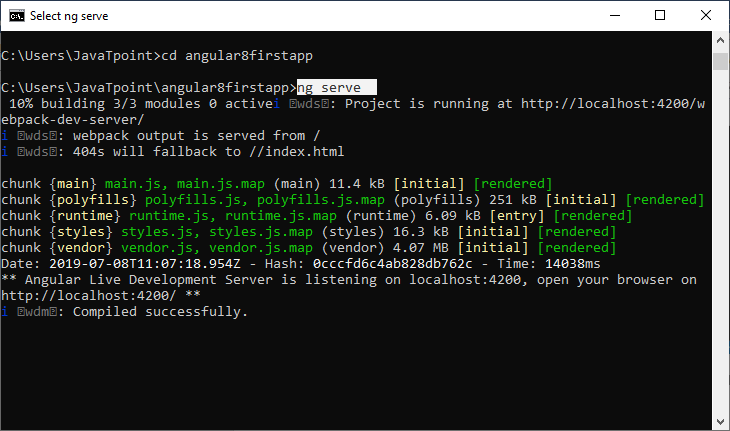


You can see that your project is open in the VS Code IDE. You can also make changes in .ts and html files to change your output accordingly.

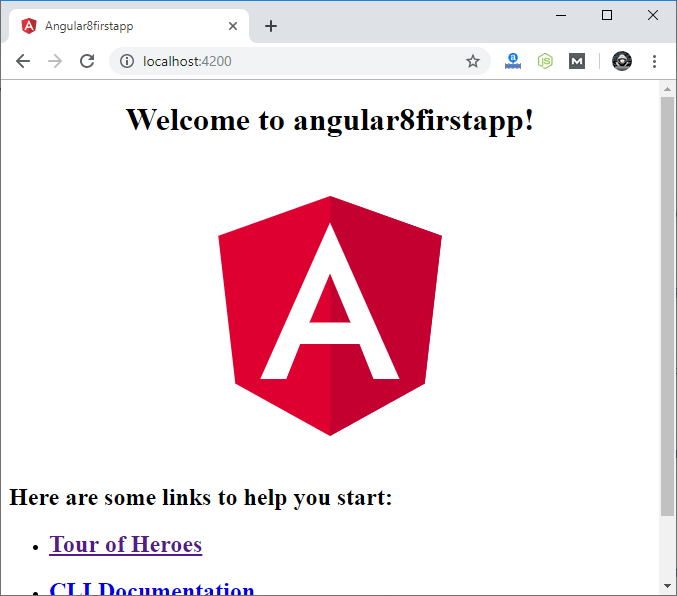
## Run your app

Open your node.js command prompt and go to your project by using cd command and then run the **ng serve** command to compile and run your app.

1. ng serve



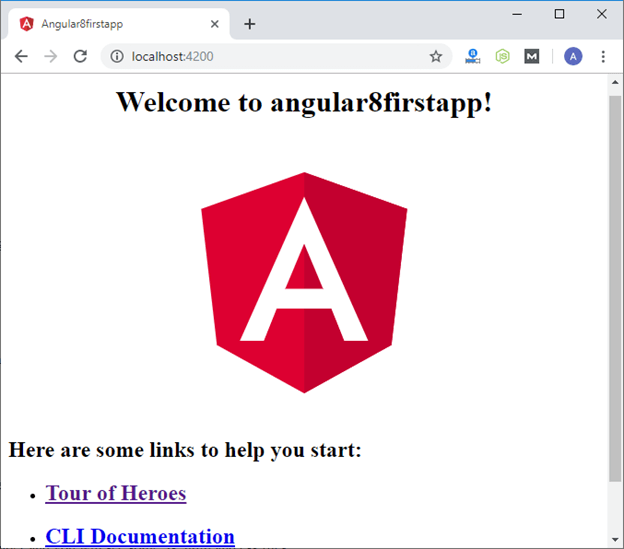
Open your browser and go to local host at: <http://localhost:4200/>



Now, you can see that your app is 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:

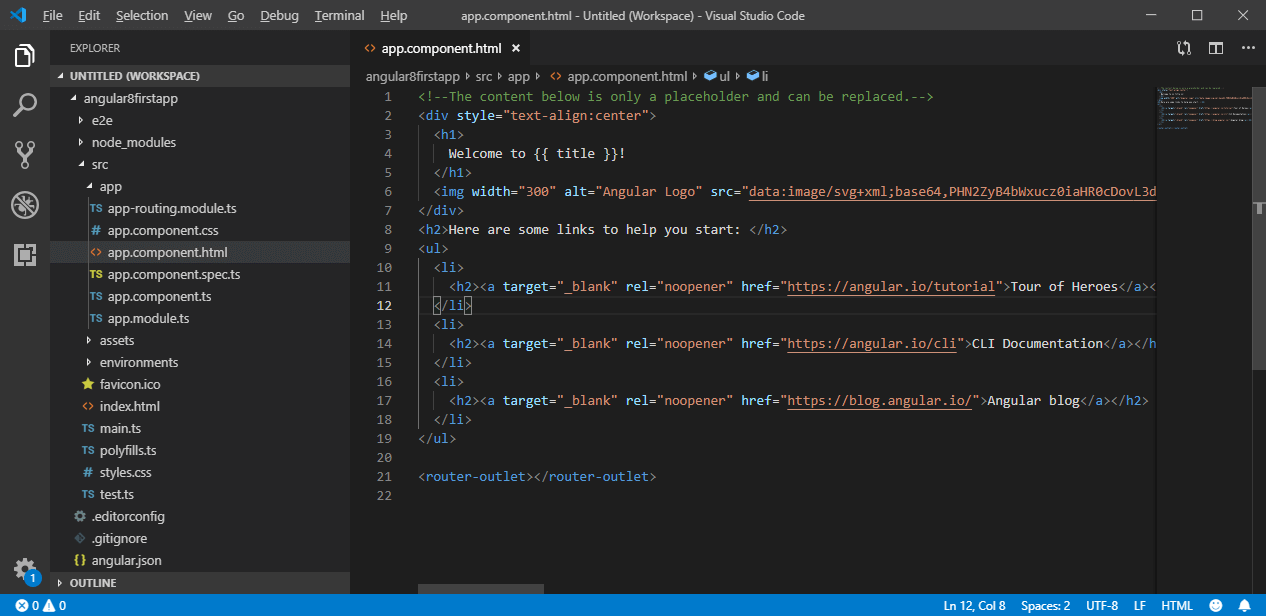
25.2M

584

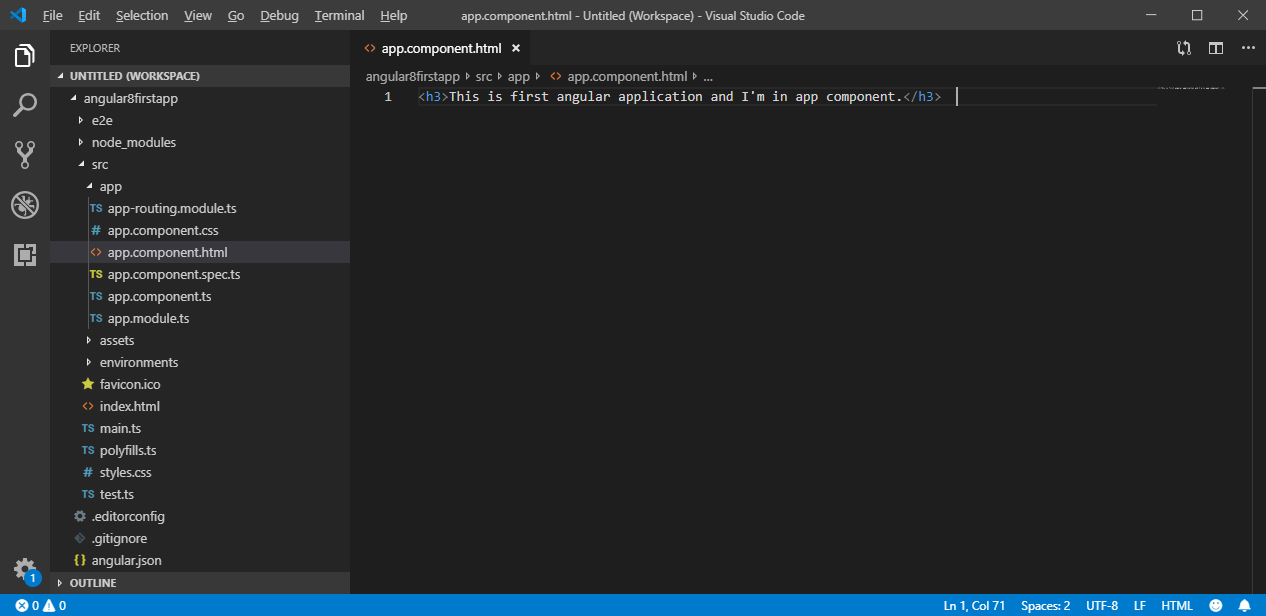
Features of Java - Javatpoint

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

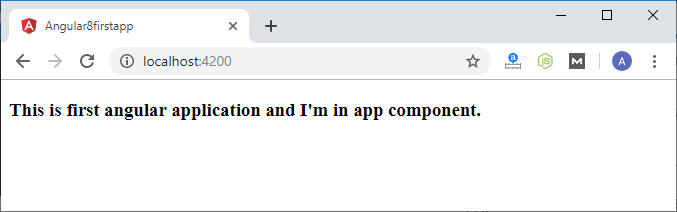
This is the original code in the app.compoment.html file



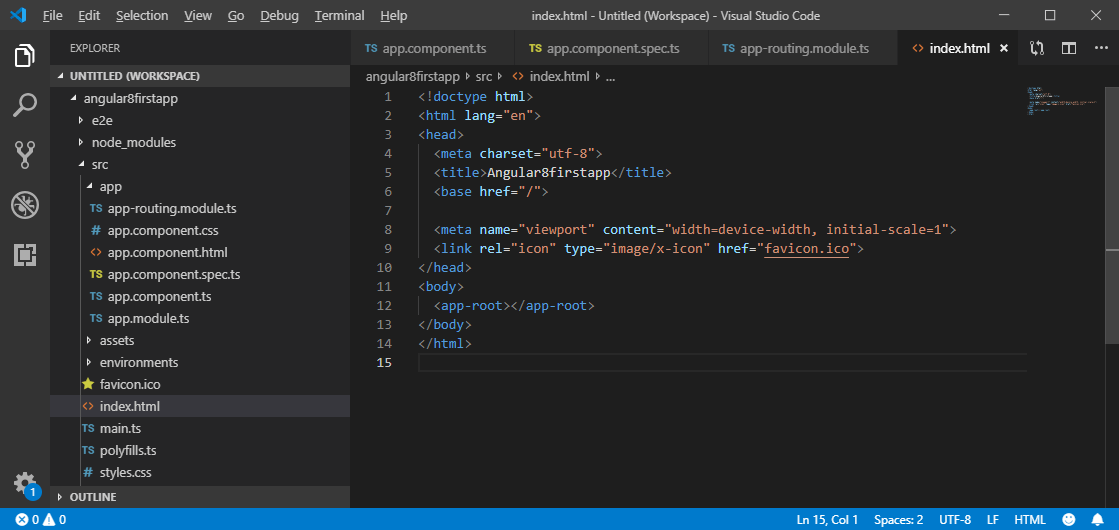
Now, it is replaced and looked like this:



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.

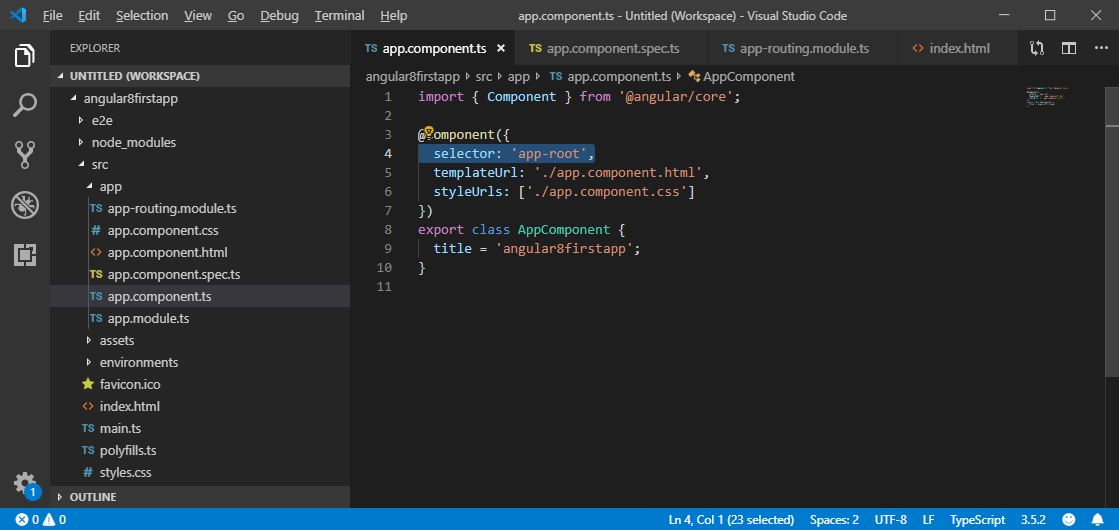
**Index.html:**

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.

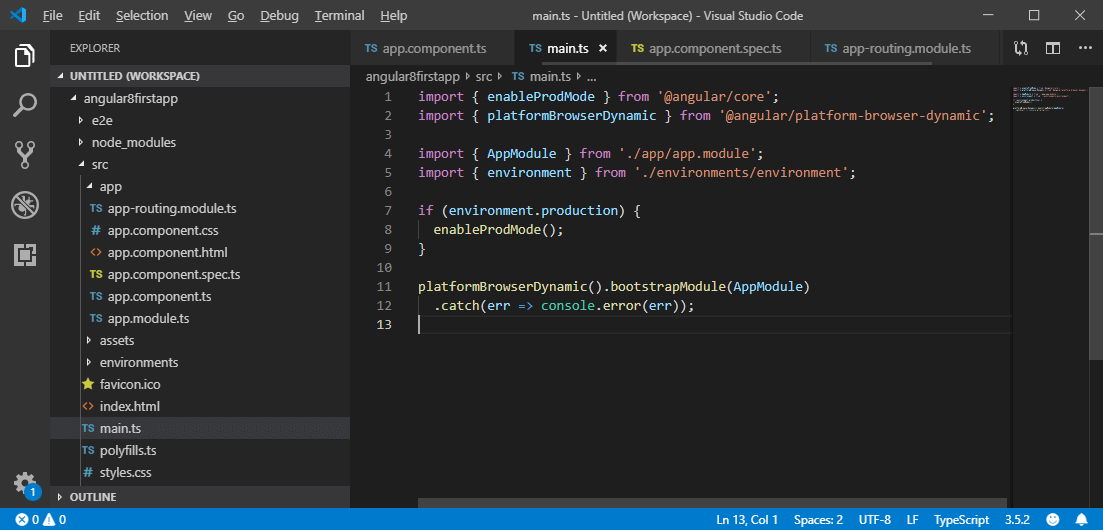
The template of the component is "./app.component.html", so, Angular includes this part into the body of the index.html file.

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 file:**

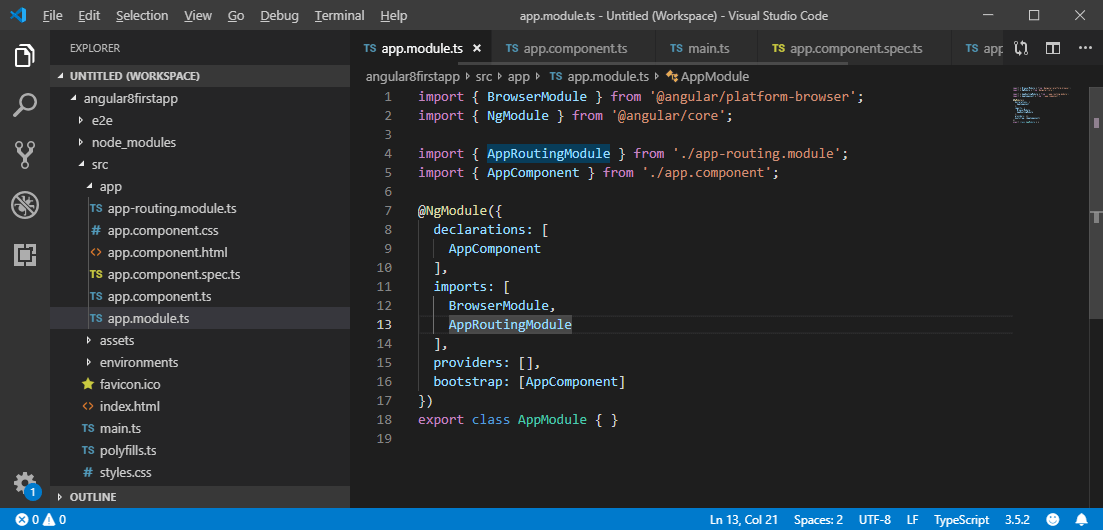
1. **import** { enableProdMode } from '@angular/core';
2. **import** { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
3. **import** { AppModule } from './app/app.module';
4. **import** { environment } from './environments/environment';
5. **if** (environment.production) {
6. enableProdMode();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



Here, the bootstrap method starts the Angular application. It refers to AppModule, which looks into the app folders. You can see in the "app.module" file that a bootstrap array which is basically a list of all the components analyzes the index.html file.

**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 { }



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

* main focus should be inside src ->app folder by default all files will be created in App folder.
* Every single page application will have index.html
* Inside index.html it has app-root its called a selector
* If u go to the app.component.ts it has selector app-root and app name.
* Go to the app-module.ts in angular application will have atleast one @ngmodule and it has to bootstrap atlease one component our case appcomponent and just double click on app component.
* Component got created in app.component.ts and got bootstrapped in app.module.ts
* @ngmodule will have all the components u created or some one created
* In imports we have to declare other angularmaterial and other imports
* First will go to index.html will have app-root so it will go to the app.component.ts
* In package.json will have all dependencies.

# Angular 8 Architecture

Angular 8 is a platform and a framework which is used to build client applications in HTML and TypeScript.

Angular 8 is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that you can import into your apps.

The basic building blocks of an Angular application are NgModules, which provide a compilation context for components. NgModules collect related code into functional sets; an Angular app is defined by a set of NgModules. An app always has at least a root module that enables bootstrapping, and typically has many more feature modules.

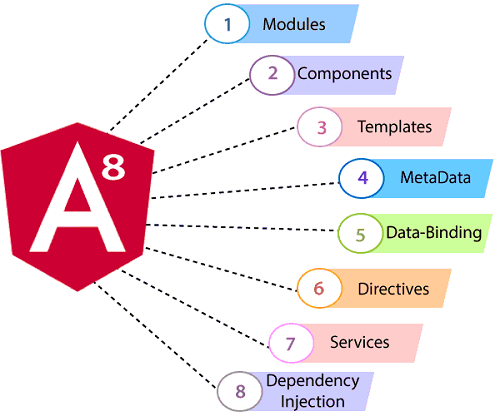
* Components define views, which are sets of screen elements that Angular can choose among and modify according to your program logic and data.
* Components use services, which provide specific functionality not directly related to views. Service providers can be injected into components as dependencies, making your code modular, reusable, and efficient.

**Key parts of Angular 8 Architecture:**

25.7M

592

Exception Handling in Java - Javatpoint



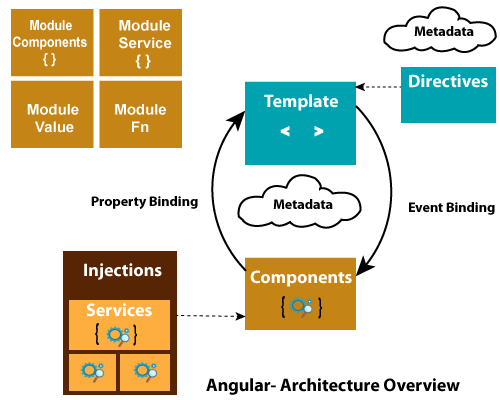
## Angular 8 Components

In Angular 8, Components and services both are simply classes with decorators that mark their types and provide metadata which guide Angular to do things.

Every Angular application always has at least one component known as root component that connects a page hierarchy with page DOM. Each component defines a class that contains application data and logic, and is associated with an HTML template that defines a view to be displayed in a target environment.

**Metadata of Component class**

* The metadata for a component class associates it with a template that defines a view. A template combines ordinary HTML with Angular directives and binding markup that allow Angular to modify the HTML before rendering it for display.
* The metadata for a service class provides the information Angular needs to make it available to components through dependency injection (DI).



## Modules

Angular 8 NgModules are different from other JavaScript modules. Every Angular 8 app has a root module known as AppModule. It provides the bootstrap mechanism that launches the application.

Generally, every Angular 8 app contains many functional modules.

**Some important features of Anngular 8 Modules:**

* Angular 8 NgModules import the functionalities form other NgModules just like other JavaScript modules.
* NgModules allow their own functionality to be exported and used by other NgModules. For example, if you want to use the router service in your app, you can import the Router NgModule.

## Template, Directives and Data Binding

In Angular 8, a template is used to combine HTML with Angular Markup and modify HTML elements before displaying them. Template directives provide program logic, and binding markup connects your application data and the DOM.

There are two types of data binding:

**1. Event Binding:** Event binding is used to bind events to your app and respond to user input in the target environment by updating your application data.

**2. Property Binding:** Property binding is used to pass data from component class and facilitates you to interpolate values that are computed from your application data into the HTML.

## Services and Dependency Injection

In Angular 8, developers create a service class for data or logic that isn't associated with a specific view, and they want to share across components.

**Dependency Injection (DI)** is used to make your component classes lean and efficient. DI doesn't fetch data from the server, validate user input, or log directly to the console; it simply renders such tasks to services.

## Routing3

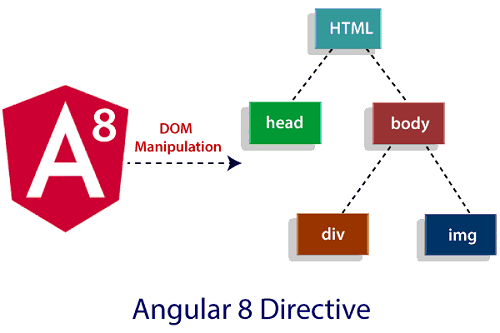
In Angular 8, Router is an NgModule which provides a service that facilitates developers to define a navigation path among the different application states and view hierarchies in their app.

**It works in the same way as a browser's navigation works. i.e.:**

* Enter a URL in the address bar and the browser will navigate to that corresponding page.
* Click the link on a page and the browser will navigate to a new page.
* Click the browser's back or forward buttons and the browser will navigate backward or forward according to your seen history pages.

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.

* **\*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.

Triggers in SQL (Hindi)

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.

# Angular 8 ngSwitch Directive

In Angular 8, ngSwitch is a structural directive which is used to Add/Remove DOM Element. It is similar to switch statement of C#. The ngSwitch directive is applied to the container element with a switch expression.

## Syntax of ngSwitch

1. **<container\_element** [ngSwitch]="switch\_expression"**>**
2. **<inner\_element** \*ngSwitchCase="match\_expresson\_1"**>**...**</inner\_element>**
3. **<inner\_element** \*ngSwitchCase="match\_expresson\_2"**>**...**</inner\_element>**
4. **<inner\_element** \*ngSwitchCase="match\_expresson\_3"**>**...**</inner\_element>**
5. **<inner\_element** \*ngSwitchDefault**>**...**</element>**
6. **</container\_element>**

## ngSwitchCase

In Angular ngSwitchCase directive, the inner elements are placed inside the container element. The ngSwitchCase directive is applied to the inner elements with a match expression. Whenever the value of the match expression matches the value of the switch expression, the corresponding inner element is added to the DOM. All other inner elements are removed from the DOM

If there is more than one match, then all the matching elements are added to the DOM.

## ngSwitchDefault

You can also apply the ngSwitchDefault directive in Angular 8. The element with ngSwitchDefault is displayed only if no match is found. The inner element with ngSwitchDefault can be placed anywhere inside the container element and not necessarily at the bottom. If you add more than one ngSwitchDefault directive, all of them are displayed.

How to find Nth Highest Salary in SQL

Any elements placed inside the container element, but outside the ngSwitchCase or ngSwitchDefault elements are displayed as it is.

## ngSwitch Directive Example

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

1. class item {
2. name: string;
3. val: number;
4. }
5. export class AppComponent
6. {
7. items: item[] = [{name: 'One', val: 1}, {name: 'Two', val: 2}, {name: 'Three', val: 3}];
8. selectedValue: string= 'One';
9. }

**Use the following code in the app.component.html file of your application:**

1. **<select** [(ngModel)]="selectedValue"**>**
2. **<option** \*ngFor="let item of items;" [value]="item.name"**>**{{item.name}}**</option>**
3. **</select>**
4. **<div** class='row' [ngSwitch]="selectedValue"**>**
5. **<div** \*ngSwitchCase="'One'"**>**One is Pressed**</div>**
6. **<div** \*ngSwitchCase="'Two'"**>**Two is Selected**</div>**
7. **<div** \*ngSwitchDefault**>**Default Option**</div>**
8. **</div>**

# 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. It is a technique to link your data to your view layer. In simple words, you can say that data binding is a communication between your typescript code of your component and your template which user sees. It makes easy to define interactive applications without worrying about pushing and pulling data.

Data binding can be either one-way data binding or two-way data binding.

## One-way databinding

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

Or

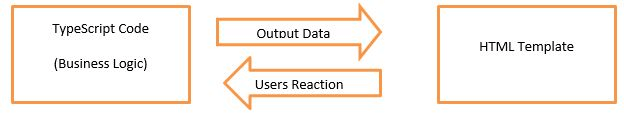
History of Java

In one-way databinding, the value of the Model is used in the View (HTML page) but you can't update Model from the View. Angular Interpolation / String Interpolation, Property Binding, and Event Binding are the example of one-way databinding.

## 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.



Angular provides four types of data binding and they are different on the way of data flowing.

* [String Interpolation](https://www.javatpoint.com/data-binding-in-angular-8#StringInterpolation)
* [Property Binding](https://www.javatpoint.com/data-binding-in-angular-8#PropertyBinding)
* [Event Binding](https://www.javatpoint.com/data-binding-in-angular-8#EventBinding)
* [Two-way binding](https://www.javatpoint.com/data-binding-in-angular-8#Two-wayBinding)

### String interpolation

String Interpolation is a **one-way databinding technique** which is used to output the data from a TypeScript code to HTML template (view). It uses the template expression in **double curly braces** to display the data from the component to the view.

**For example:**

{{ data }}

String interpolation adds the value of a property from the component:

**Syntax:**

1. **<li>**Name: {{ user.name }}**</li>**
2. **<li>**Email: {{ user.email }}**</li>**

Learn more about String Interpolation: [**Click Here**](https://www.javatpoint.com/string-interpolation-in-angular-8)

### Property Binding

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.

**For example:**

<img [src]="imgUrl"/>

**Syntax:**

1. **<input** type="email" [value]="user.email"**>**

Learn more about Property Binding: [**Click Here**](https://www.javatpoint.com/property-binding-in-angular-8)

### Event Binding

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>**

Learn more about Event Binding: [**Click Here**](https://www.javatpoint.com/event-binding-in-angular-8)

### Two-way Data Binding

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 data binding, property binding and event binding are combined together.

**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.

Learn more about two way Data Binding: [**Click Here**](https://www.javatpoint.com/two-way-data-binding-in-angular-8)

# 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" />

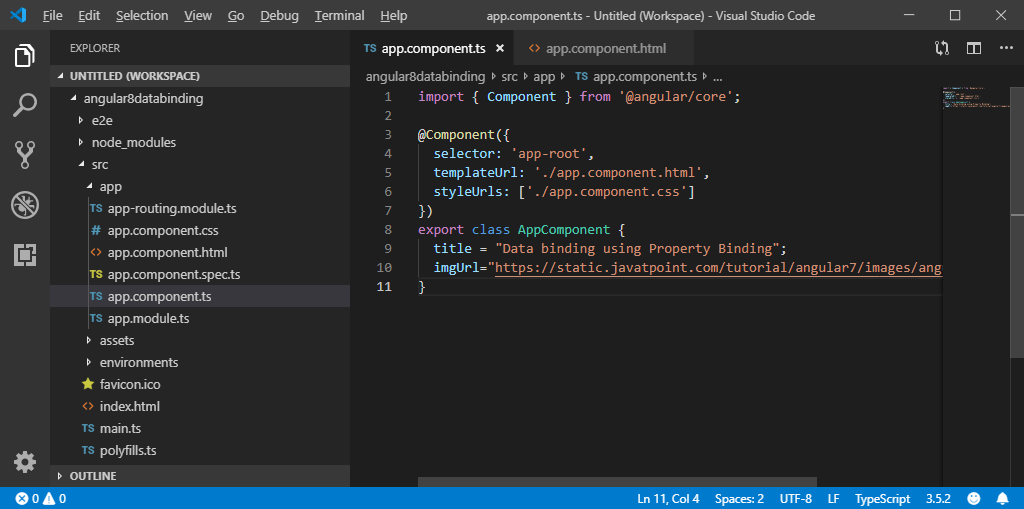
Property binding is preferred over string interpolation because it has shorter and cleaner code String interpolation should be used when you want to simply display some dynamic data from a component on the view between headings like h1, h2, p etc.

#### 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

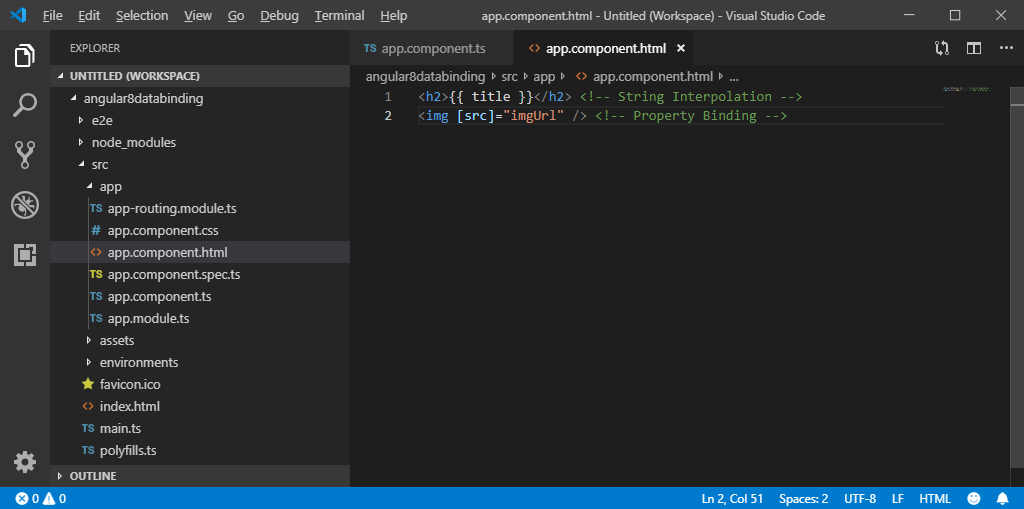
Open **app.componnt.ts** file and add the following code:

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. }



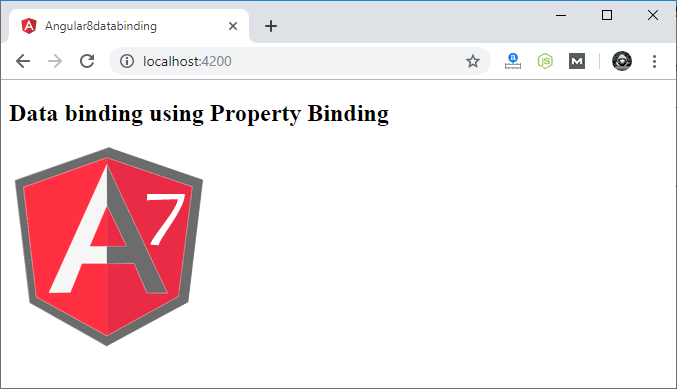
Now, open **app.component.html** and use the following code for property binding:

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:**



# String Interpolation in Angular 8

String Interpolation is a **one-way databinding** technique which is used to output the data from a TypeScript code to HTML template (view). It uses the template expression in **double curly braces** to display the data from the component to the view. String interpolation adds the value of a property from the component.

**For example:**

{{ data }}

We have already created an Angular project using Angular CLI.

20.1M

427

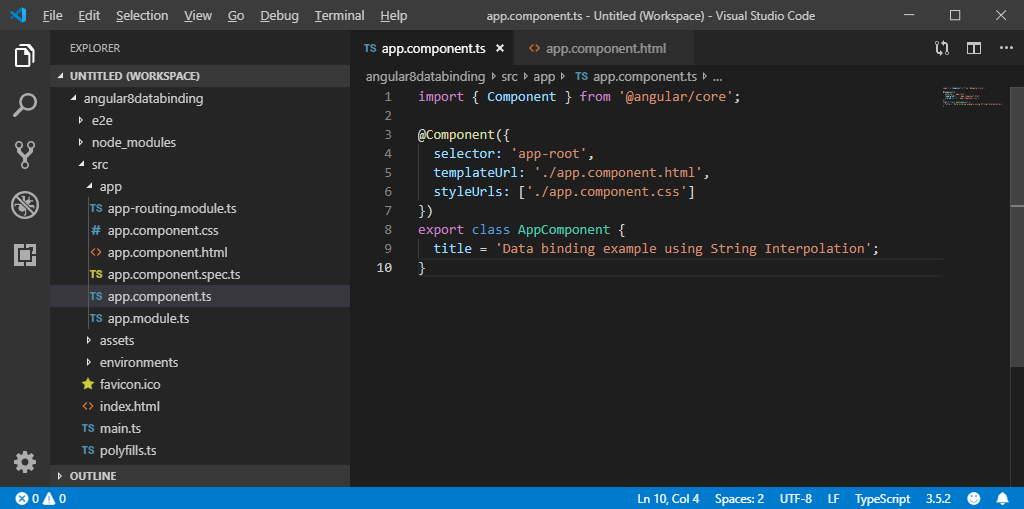
HTML Tutorial

**See:** How to create Angular 8 project. [Click Here](https://www.javatpoint.com/angular-8-first-app)

Here, we are using the same project for this example.

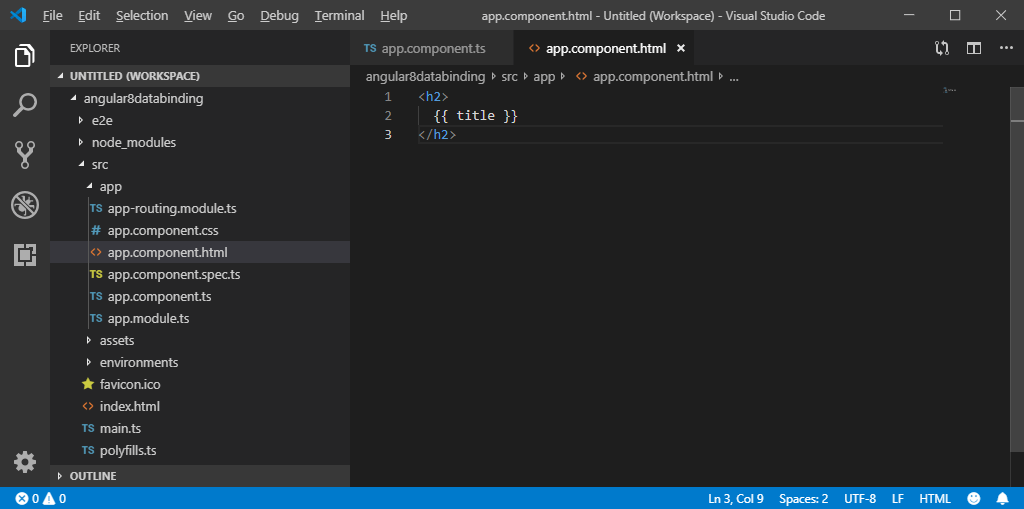
Open **app.component.ts** file and use the following code within the file:

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. title = 'Data binding example using String Interpolation';
9. }



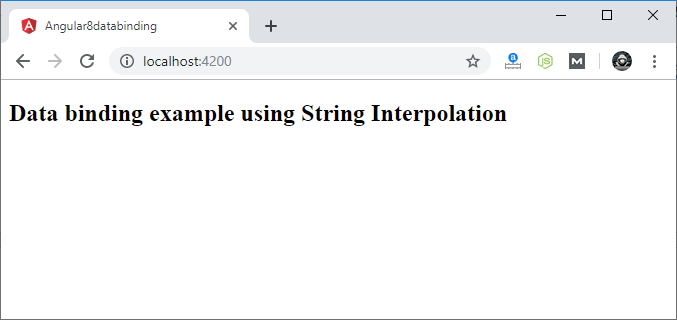
Now, open **app.component.html** and use the following code to see string interpolation.

1. **<h2>**
2. {{ title }}
3. **</h2>**



Now, open Node.js command prompt and run the **ng serve** command to see the result.

**Output:**



String Interpolation can be used to resolve some other expressions too. Let's see an example.

### Example:

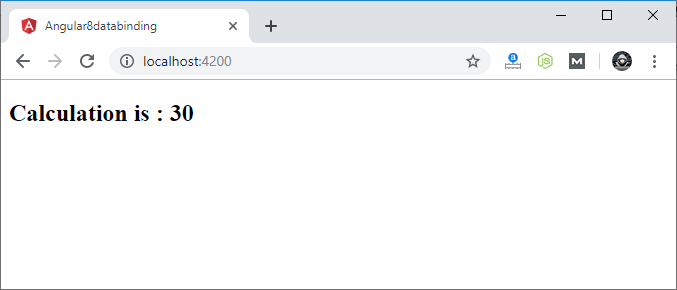
Update the **app.component.ts** file with the following code:

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 example using String Interpolation';
9. numberA: number = 10;
10. numberB: number = 20;
11. }

**app.component.html:**

1. **<h2>**Calculation is : {{ numberA + numberB }}**</h2>**

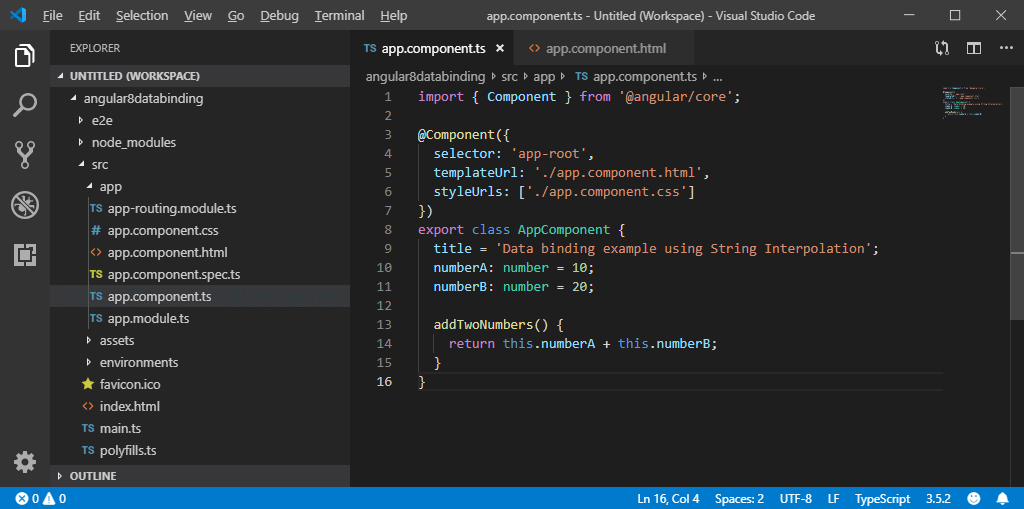
**Output:**



You can use the same application in another way:

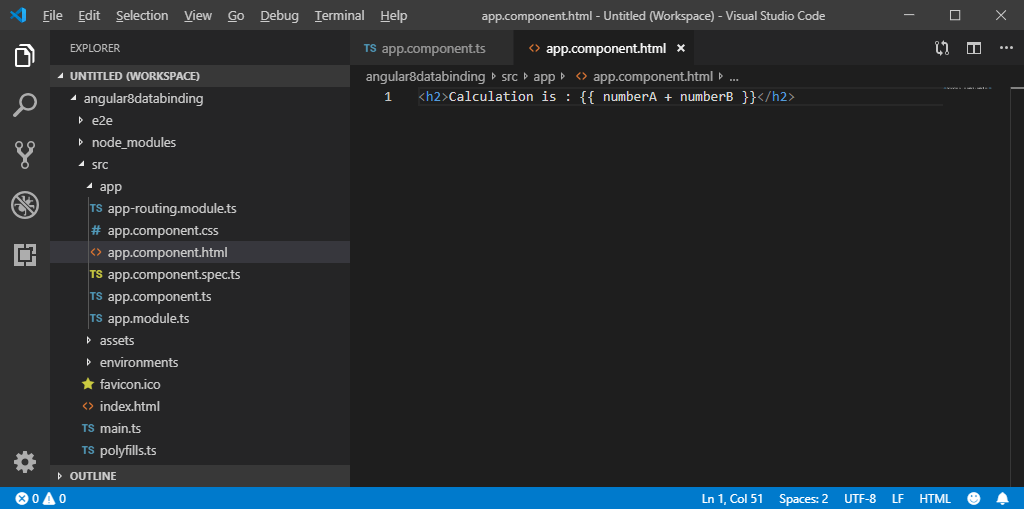
**App.component.ts:**

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 example using String Interpolation';
9. numberA: number = 10;
10. numberB: number = 20;
11. addTwoNumbers() {
12. return this.numberA + this.numberB;
13. }
14. }

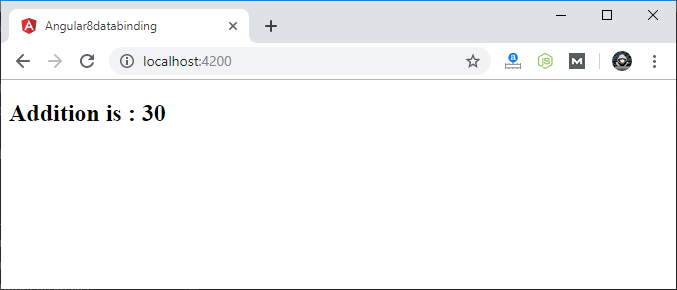


**App.component.html:**

1. **<h2>**Calculation is : {{ numberA + numberB }}**</h2>**



**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>**

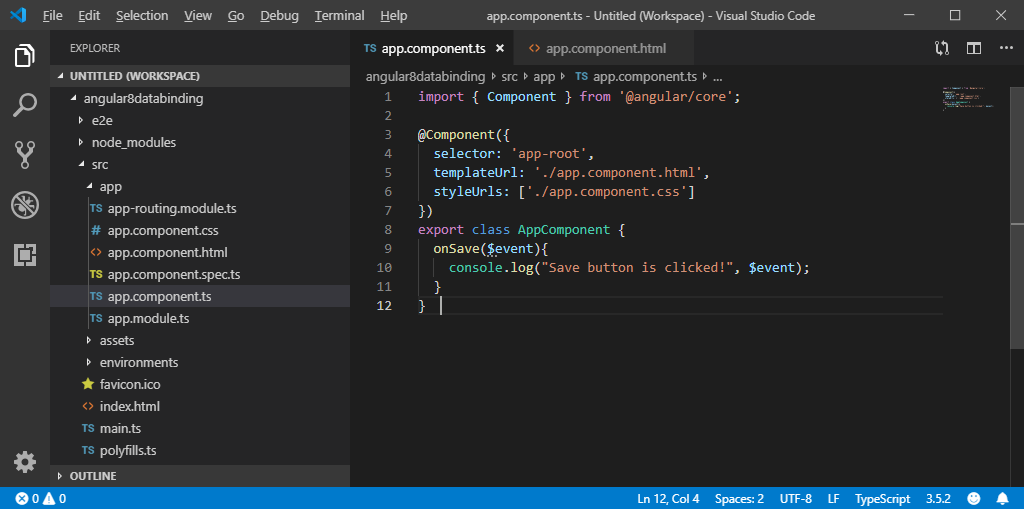
### Event Binding Example

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:

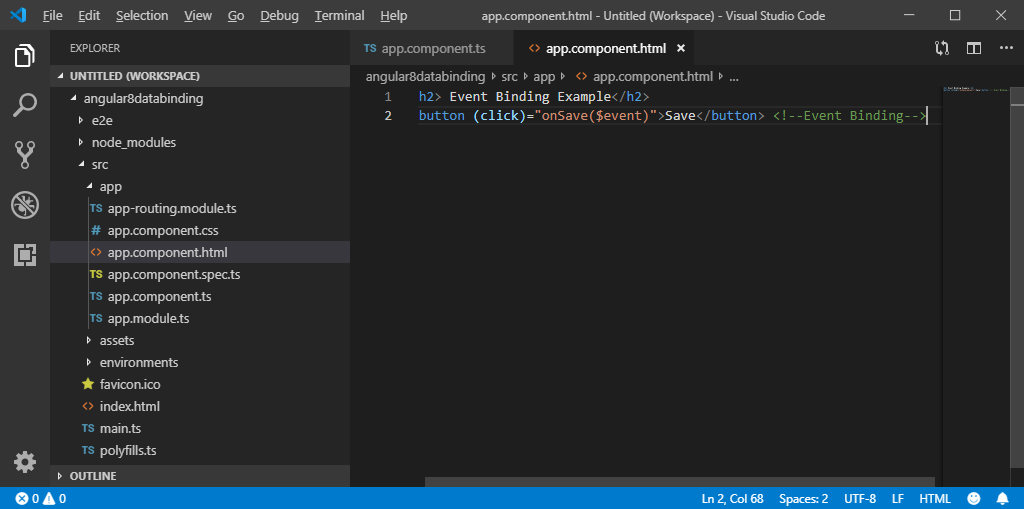
How to find Nth Highest Salary in SQL

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. }

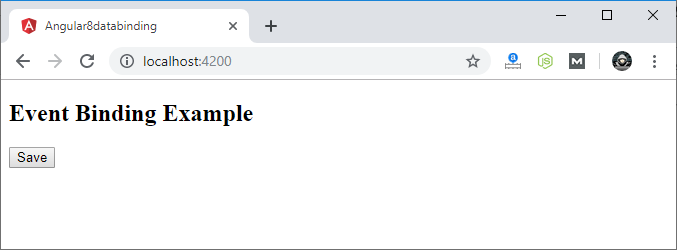


**app.component.html:**

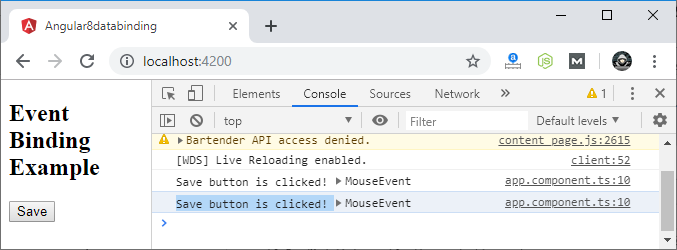
1. **<h2>** Event Binding Example**</h2>**
2. **<button** (click)="onSave($event)"**>**Save**</button>** <!--Event Binding-->



**Output:**



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



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

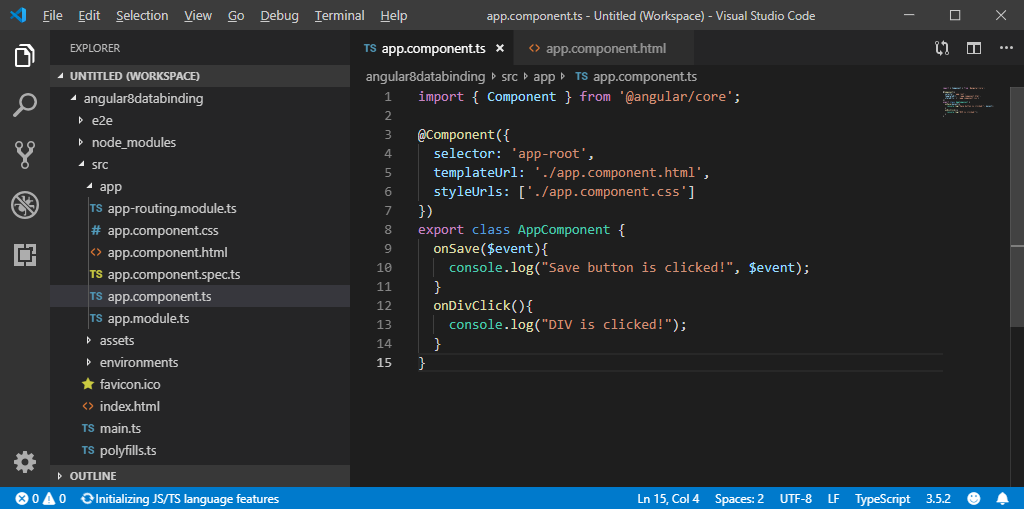
## Event Bubbling

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.

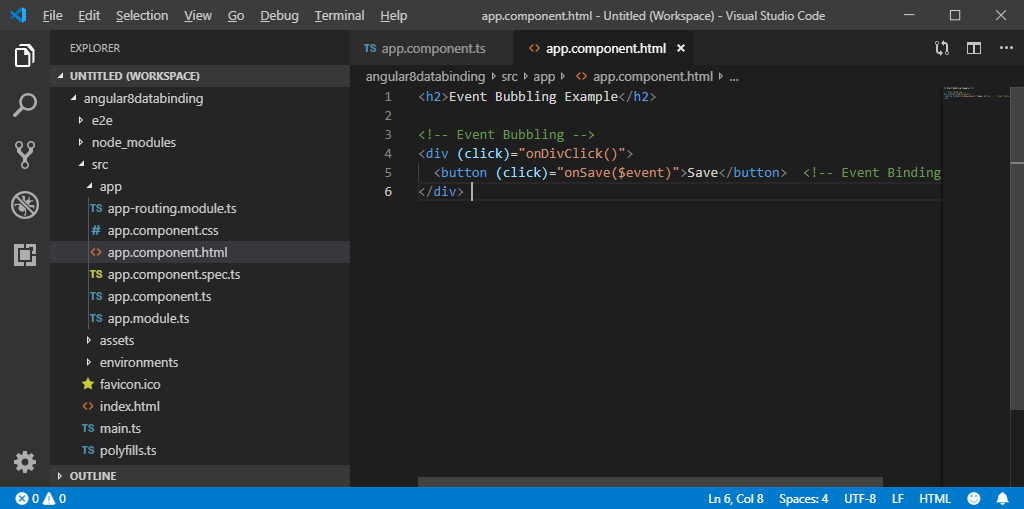
Use the following code in **app.component.ts** file:

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:**

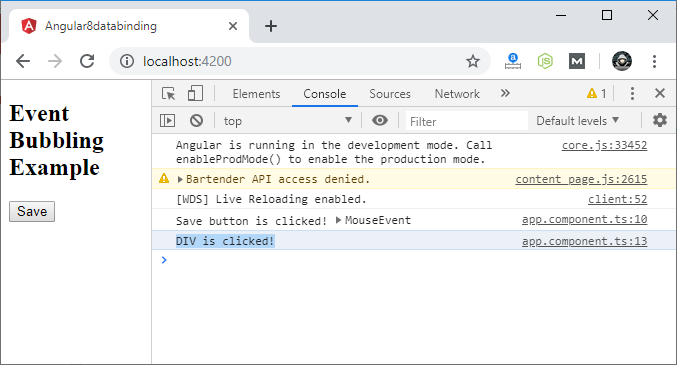
1. **<h2>**Event Bubbling Example**</h2>**
2. <!-- Event Bubbling -->
3. **<div** (click)="onDivClick()"**>**
4. **<button** (click)="onSave($event)"**>**Save**</button>**  <!-- Event Binding -->
5. **</div>**



**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.

[**Next →**](https://www.javatpoint.com/angular-vs-react)[**← Prev**](https://www.javatpoint.com/two-way-data-binding-in-angular-8)

# 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.

10 Sec

24.4M

544

C++ vs Java

### 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.

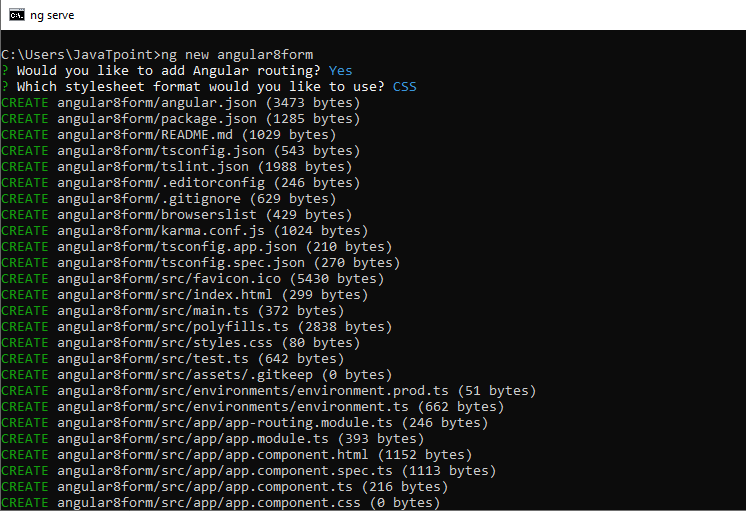
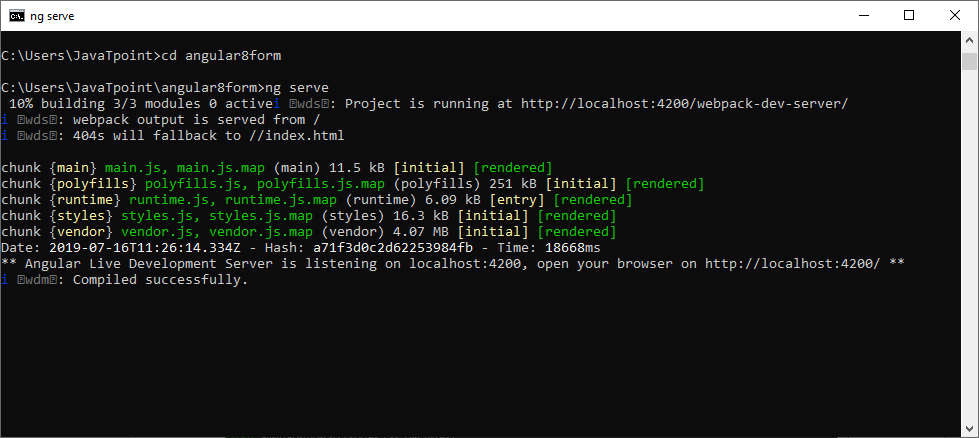
## Angular 8 Form Example

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

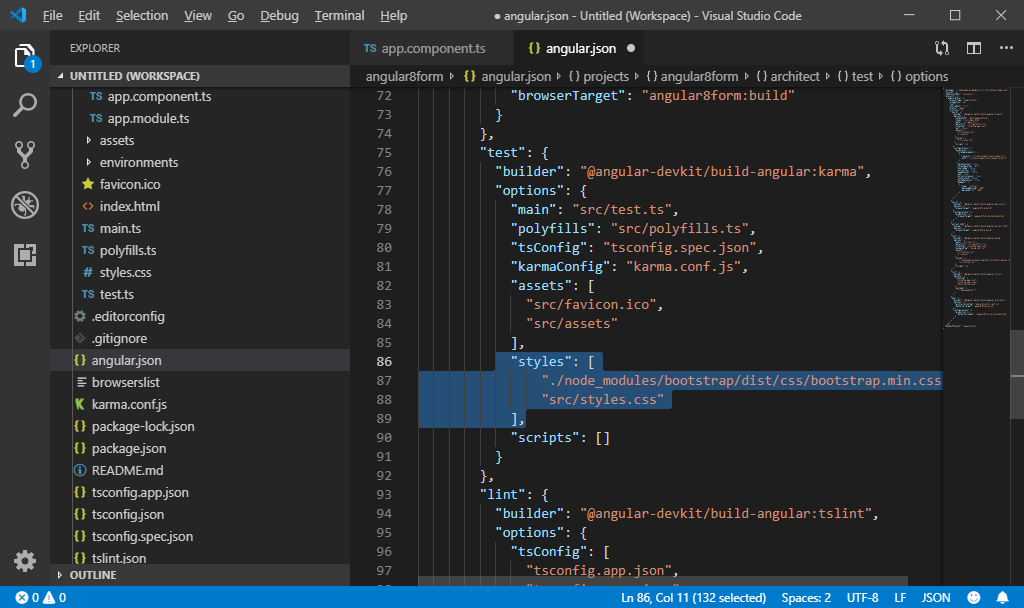
  


* Install the Bootstrap 4 using the following command.

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. ],

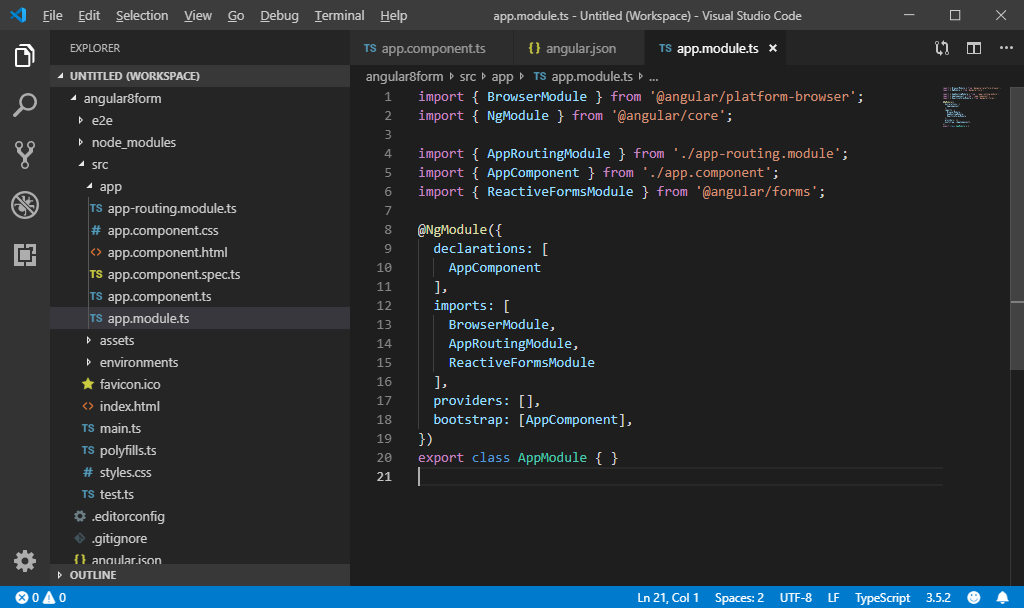


* Register the Reactive Forms Module

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 { }

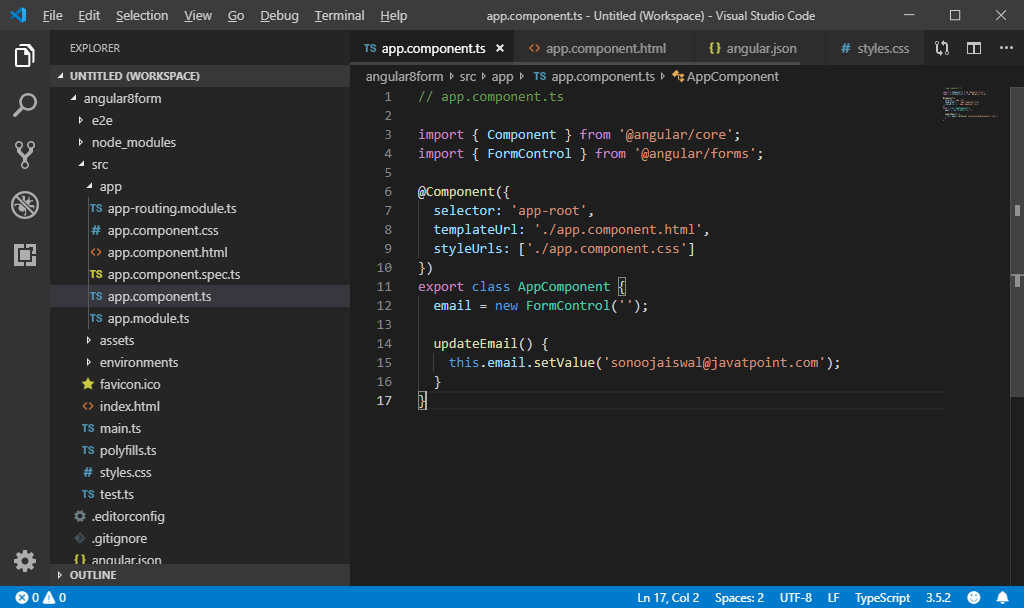


* Add FormControl class register the control into the template and update the FormControl value

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.

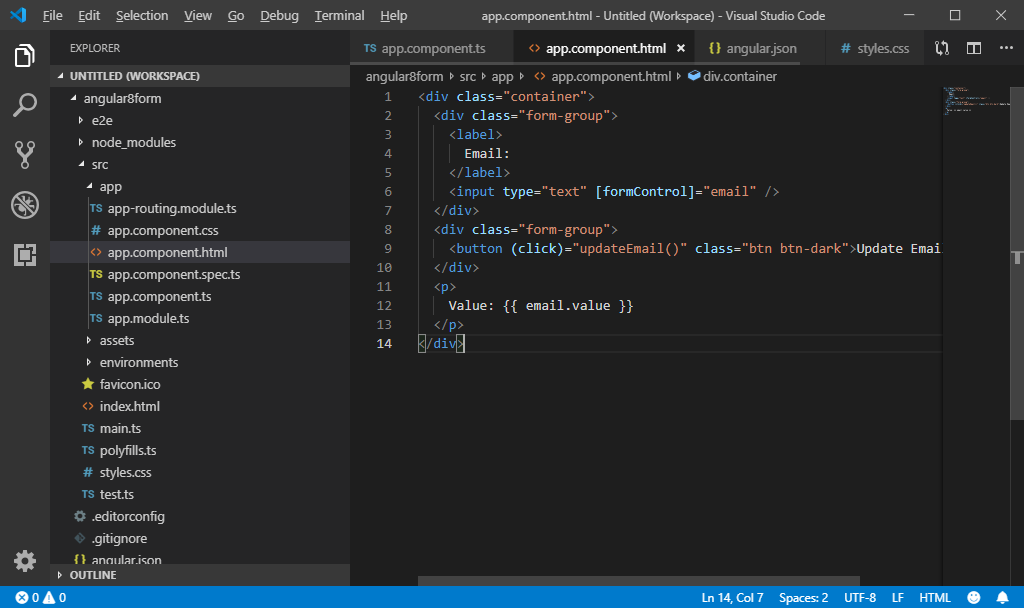
Now, modify the **app.component.ts** file.

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. }



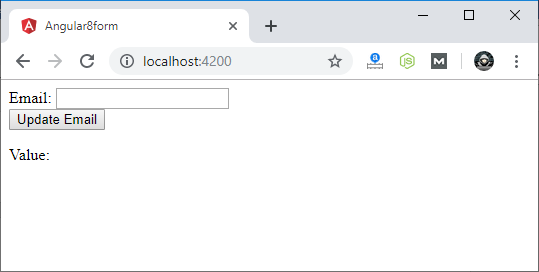
Also, update the view **app.component.html** file.

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>**

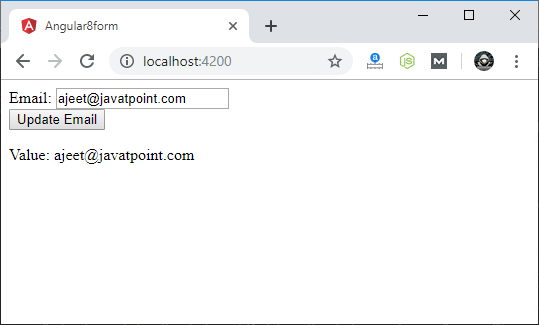


Now, save your code and start the server.

**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.

