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.

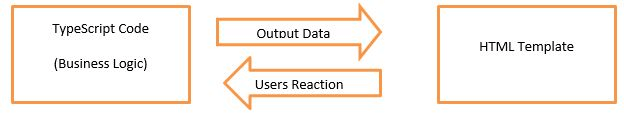
Or

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.

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" />

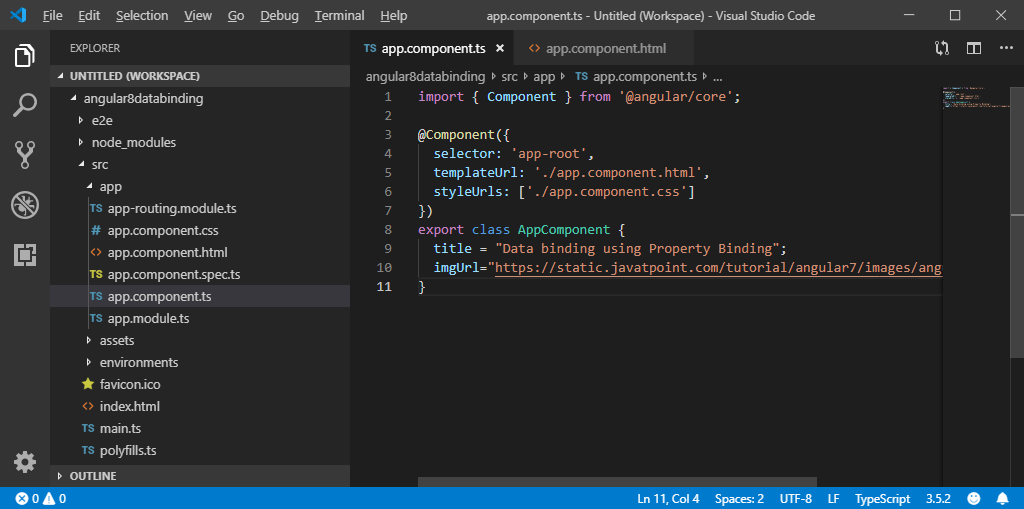
Property binding is preferred over string interpolation because it has shorter and cleaner code String interpolation should be used when you want to simply display some dynamic data from a component on the view between headings like h1, h2, p etc.

**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

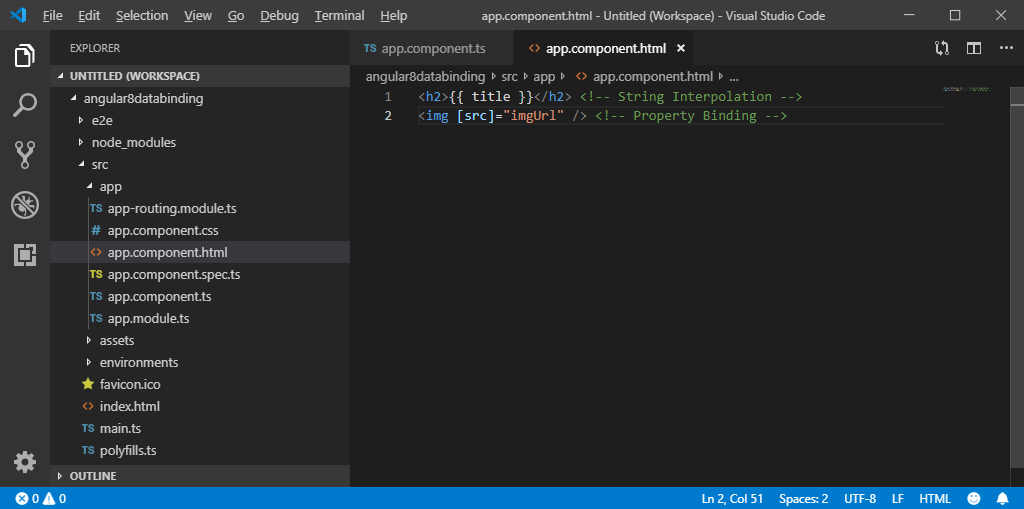
Open **app.componnt.ts** file and add the following code:

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. }



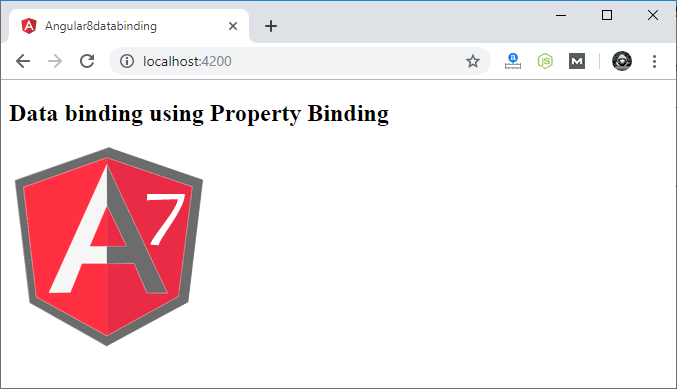
Now, open **app.component.html** and use the following code for property binding:

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 }}

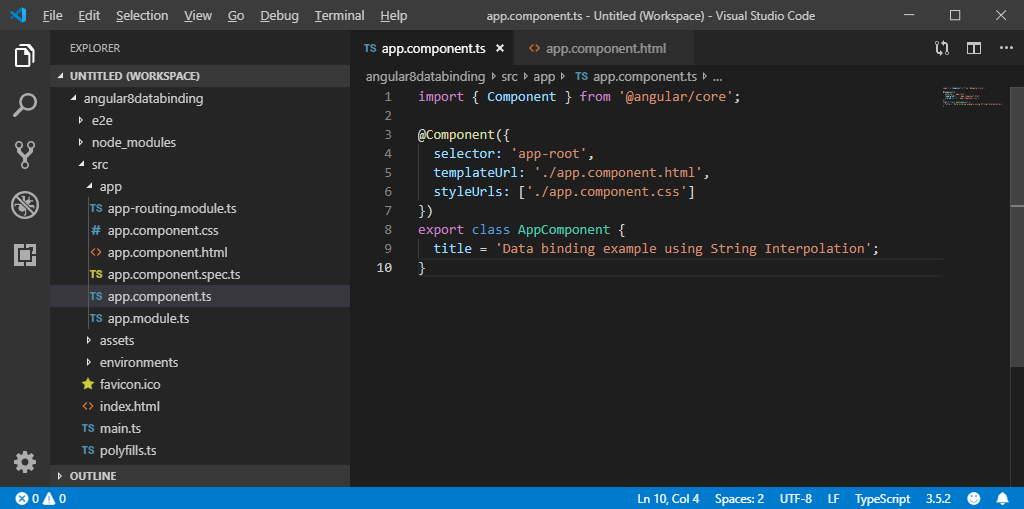
We have already created an Angular project using Angular CLI.

**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.

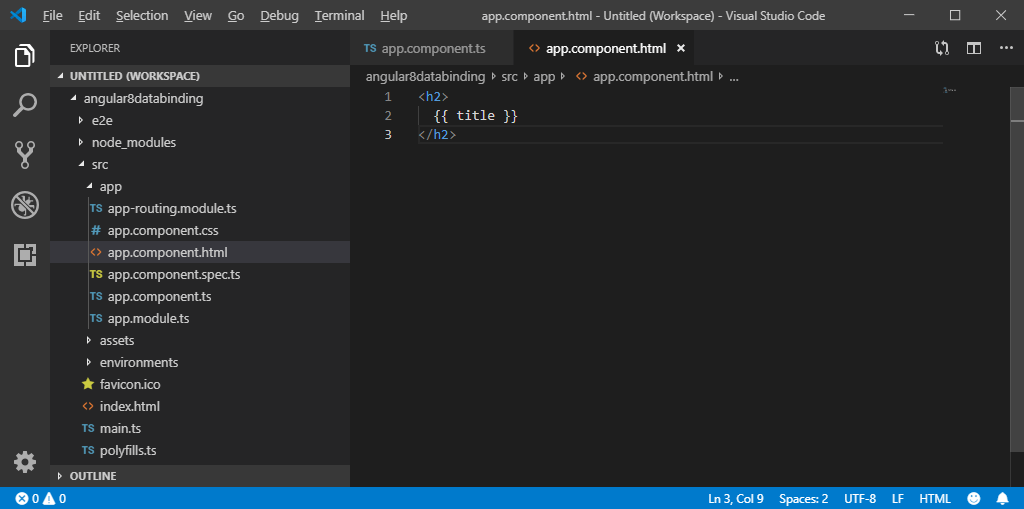
Open **app.component.ts** file and use the following code within the file:

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. }



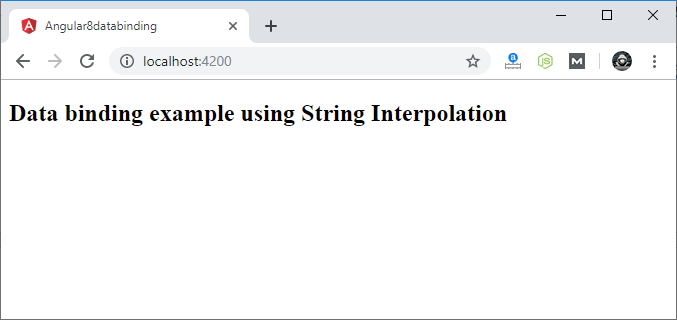
Now, open **app.component.html** and use the following code to see string interpolation.

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:

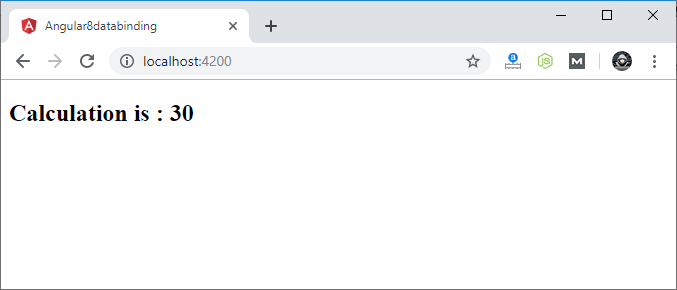
Update the **app.component.ts** file with the following code:

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>**

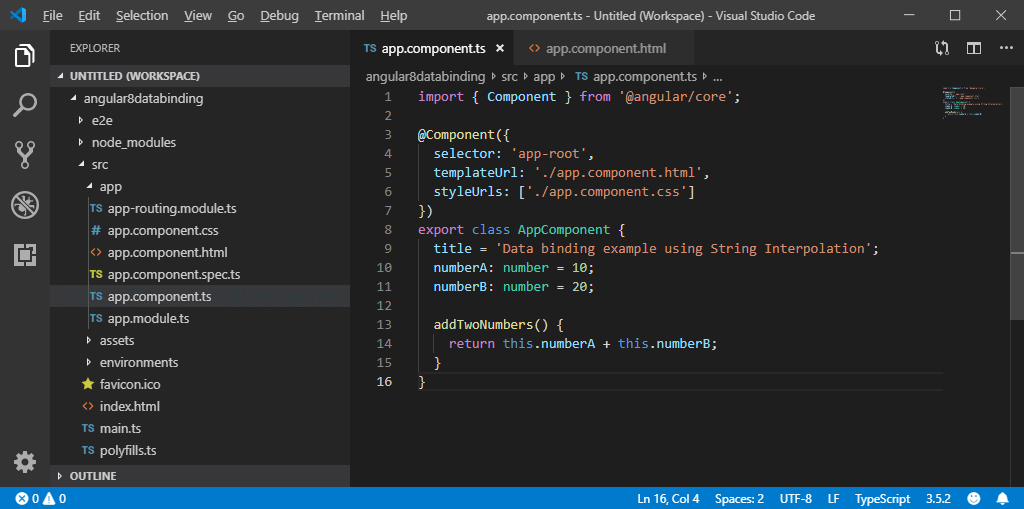
**Output:**



You can use the same application in another way:

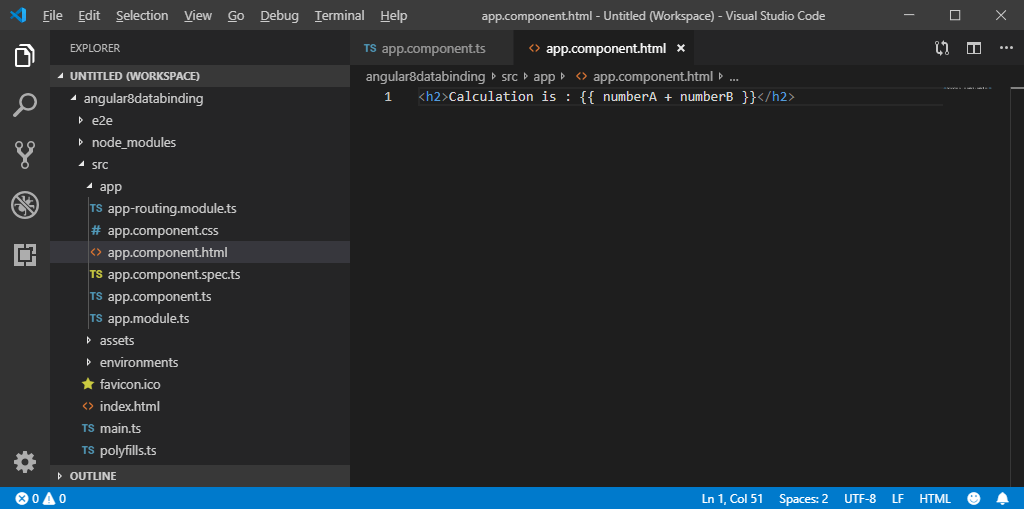
**App.component.ts:**

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. }

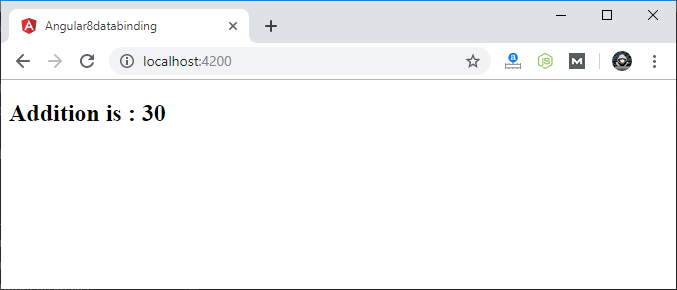


**App.component.html:**

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



**Output:**



# 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.

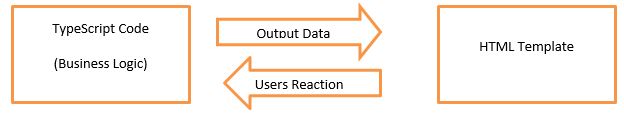
This happens immediately and automatically, ensures that the HTML template and the TypeScript code are updated at all times.

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

### Syntax:

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

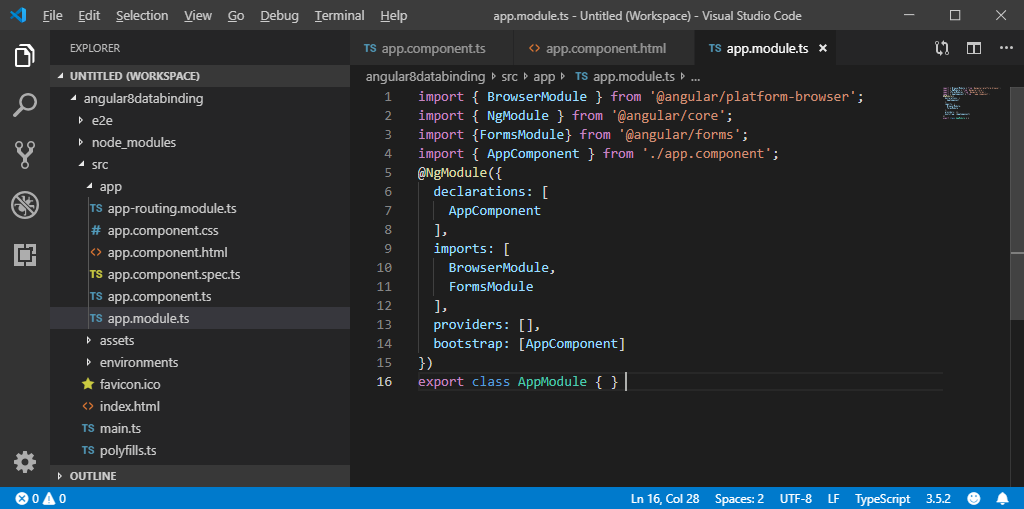
#### Note: For two way data binding, we have to enable the ngModel directive. It depends upon FormsModule in angular/forms package, so we have to add FormsModule in imports[] array in the AppModule.



Let's take an example to understand it better.

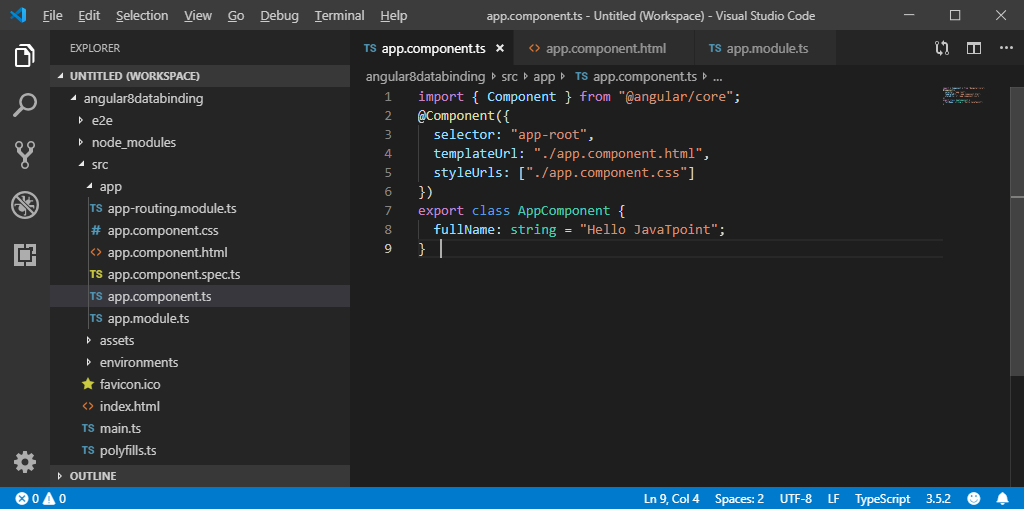
Open your project's **app.module.ts** file and use the following code:

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 { }



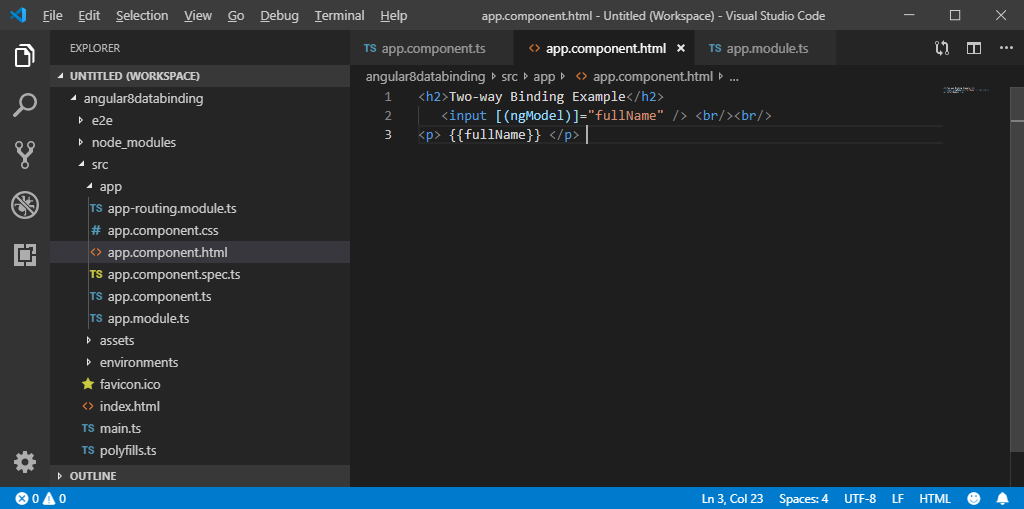
**app.component.ts file:**

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. }



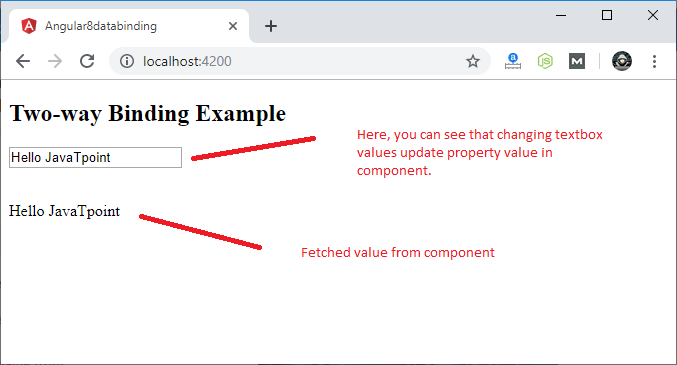
**app.component.html file:**

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:**

