Module 2 – Angular Components And Databinding

Demo Document 2: Types Of Data Binding

edureka!

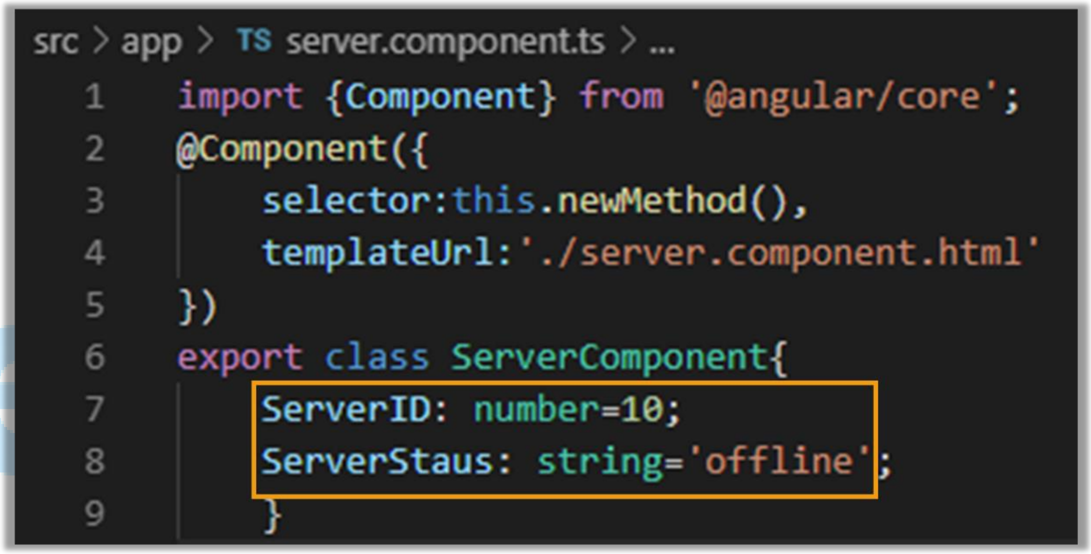
In this demo, we will see how to make use of Data Binding in an Angular application

String Interpolation Example

Step 1 – Inside the 'server.component.ts' file, create two properties- **'ServerID'** and **'ServerStatus'** and assign values to it

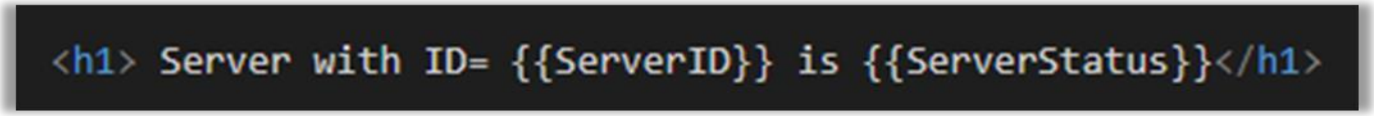
Step 2 – Bind **'ServerID'** and **'ServerStatus'** to HTML and display the message "Server with ID = ___ is ___"

The typescript data to be inserted in HTML



```
import {Component} from '@angular/core';
@Component({
    selector:this.newMethod(),
    templateUrl:'./server.component.html'
})
export class ServerComponent{
    ServerID: number=10;
    ServerStaus: string='offline';
    }
```

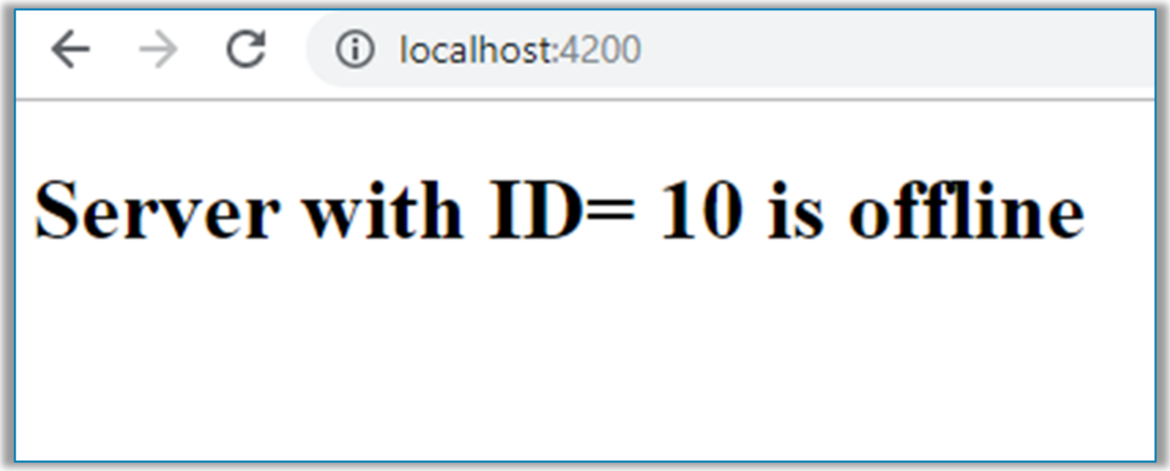
Step3 – The "ServerID" and "ServerStatus" are bound to the template as following



```
<h1> Server with ID= {{ServerID}} is {{ServerStatus}}</h1>
```

Property that is bound in between the curly braces must always return a string

Step 4 – On localhost:4200 you will see the following output

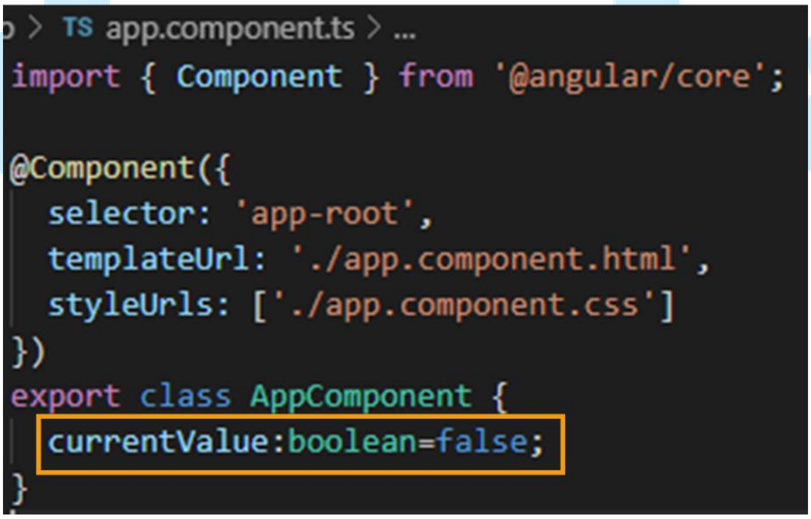


Server with ID= 10 is offline

Difference between String Interpolation and Property Binding

Step 1 – Inside the component file, create a Boolean value named ‘currentValue’ and assign ‘false’ value to it

app.component.ts (the typescript file)



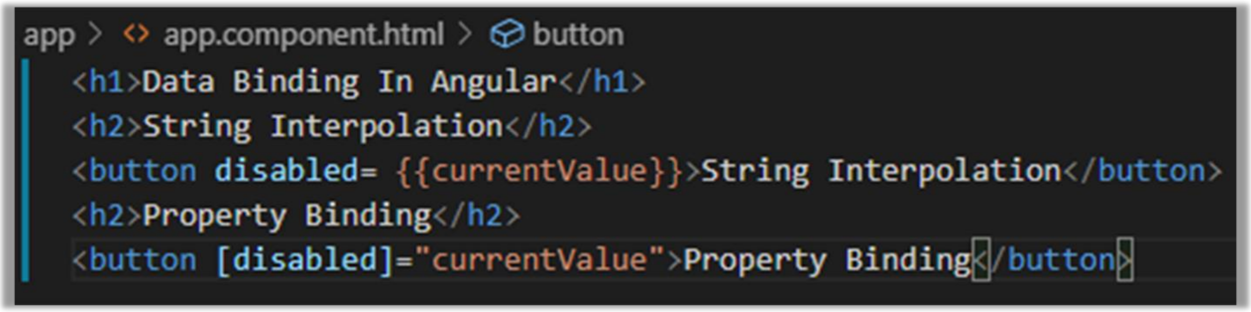
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  currentValue:boolean=false;
}
```

Step 2 – Bind this value to the html property - ‘disabled’

Step 3 –

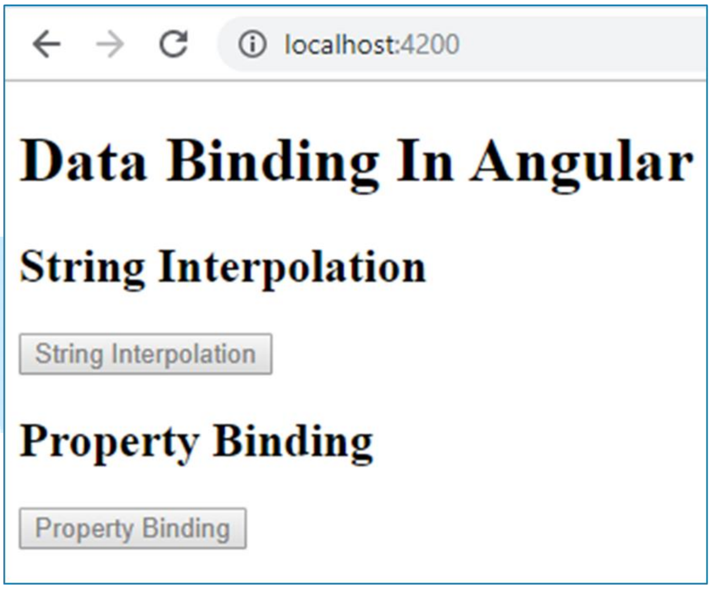
- Create two buttons: String interpolation and Property Binding
- For String Interpolation enclose the ‘currentValue’ in curly braces and attach it to ‘disabled’ property
- For Property Binding enclose the ‘disabled’ property in square brackets and assign ‘currentValue’ to it

app.component.html file

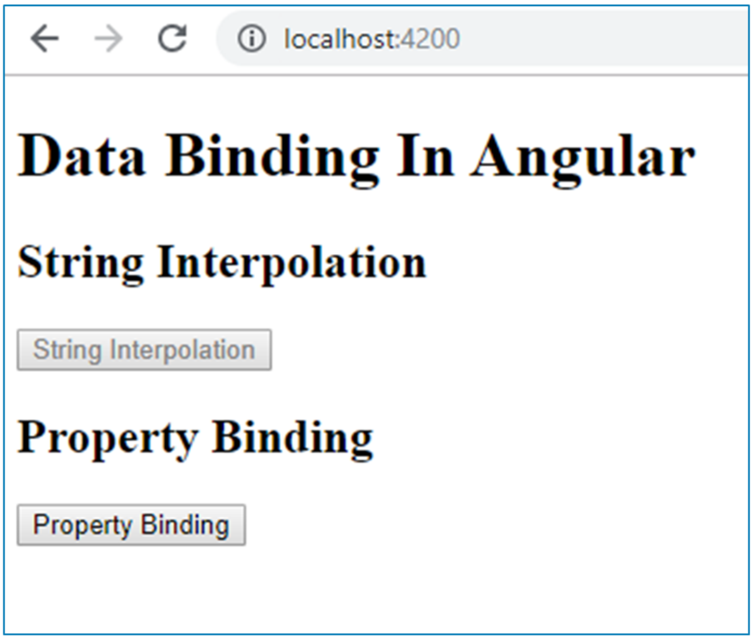


```
app > <> app.component.html > button
    <h1>Data Binding In Angular</h1>
    <h2>String Interpolation</h2>
    <button disabled= {{currentValue}}>String Interpolation</button>
    <h2>Property Binding</h2>
    <button [disabled]="currentValue">Property Binding</button>
```

Step 4 – As we can see, for String Interpolation the button remains disabled irrespective of the value assigned to ‘currentValue’



When ‘currentValue’ is false, button is disabled



Data Binding In Angular

String Interpolation

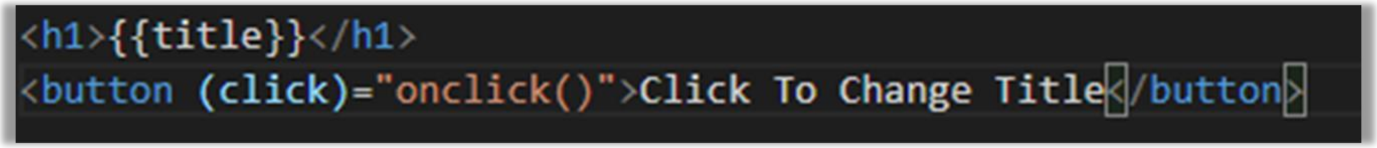
Property Binding

When 'currentValue' is true, button is enabled only for property binding

Event Binding Example

Step 1 – In our example we want to change the title of the page when a button is clicked. We enclose this click event within parenthesis and assign the corresponding event handler

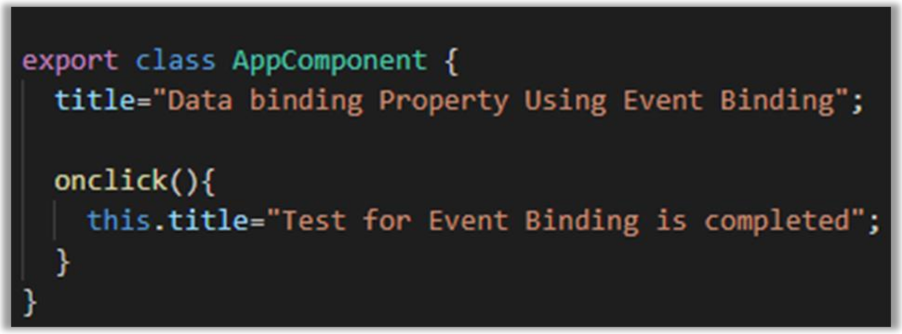
app.component.html File



```
<h1>{{title}}</h1>
<button (click)="onclick()">Click To Change Title</button>
```

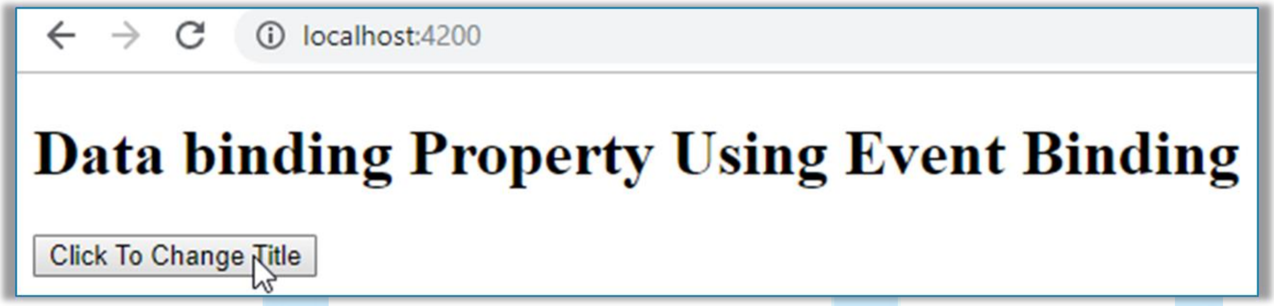
Step 2 – Assign the event handler 'onclick' and define it in the component class. In the "onclick" method we are going to change the title of the page when the button is clicked

app.component.ts File

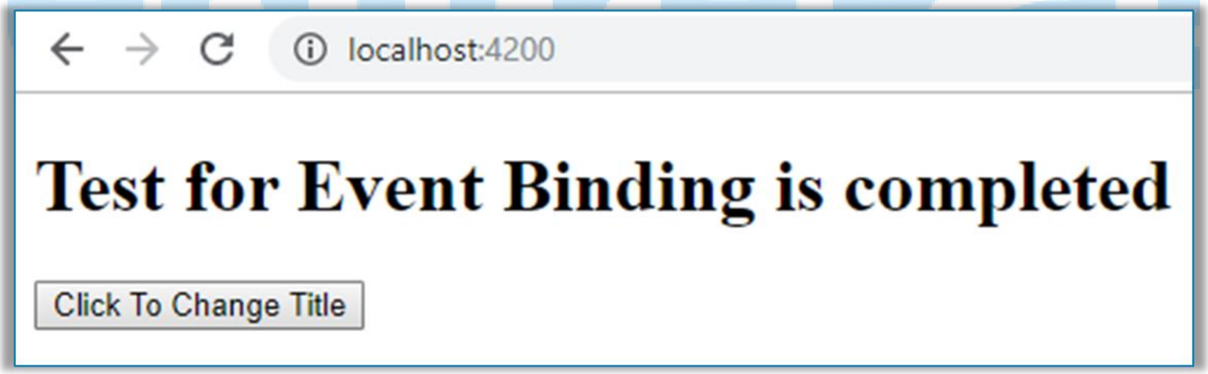


```
export class AppComponent {
  title="Data binding Property Using Event Binding";

  onclick(){
    this.title="Test for Event Binding is completed";
  }
}
```

Step 11 – Output



Data binding Property Using Event Binding

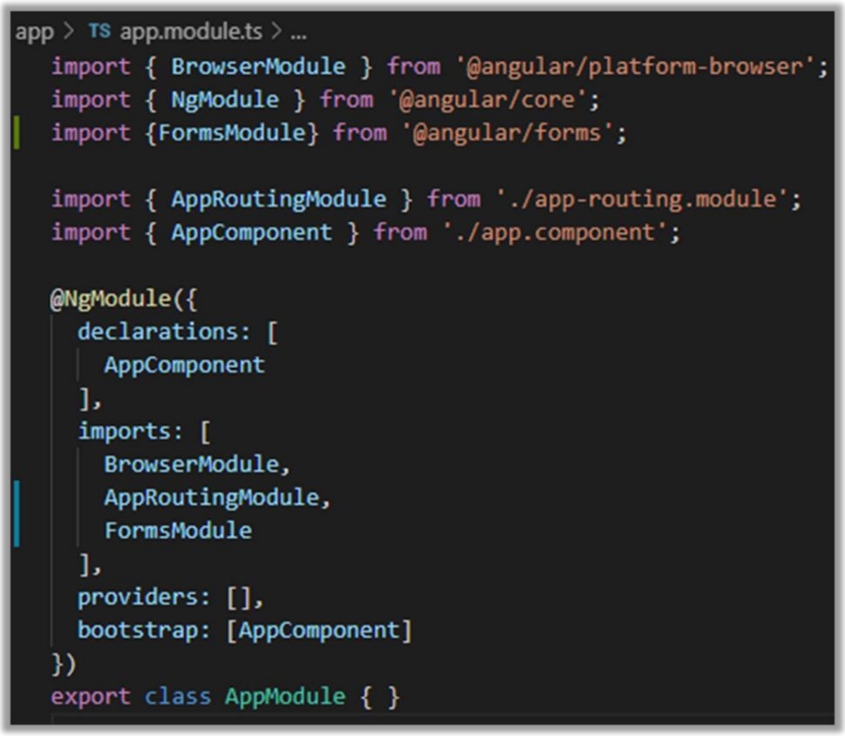


Test for Event Binding is completed

Two-Way Binding Example

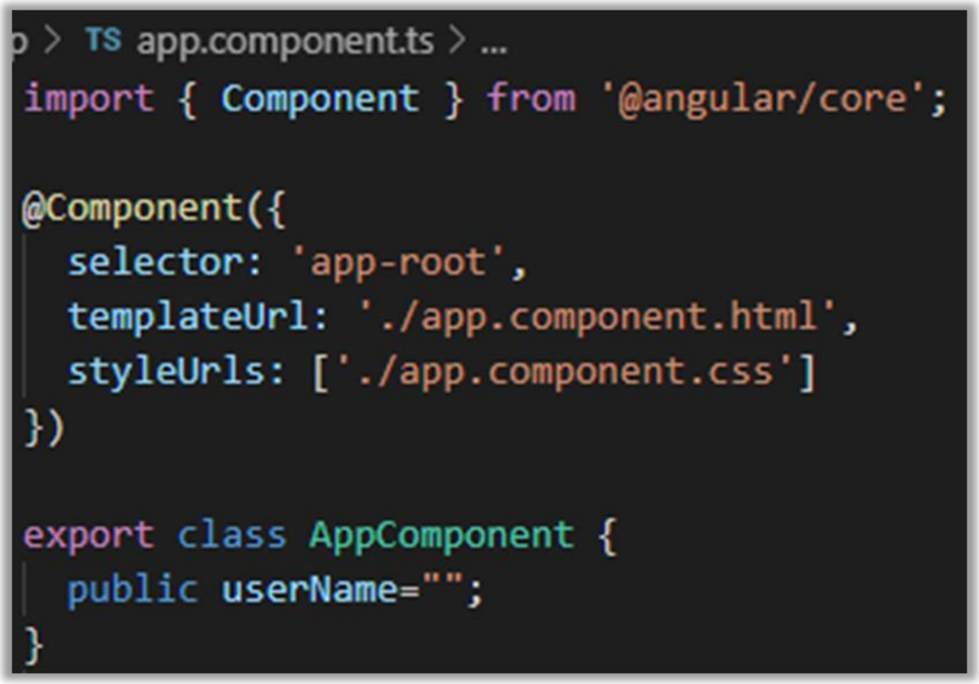
Step 1 –

- In app.module.ts import FormsModule library from ‘@angular/forms’ as ngModel is defined in the forms module library

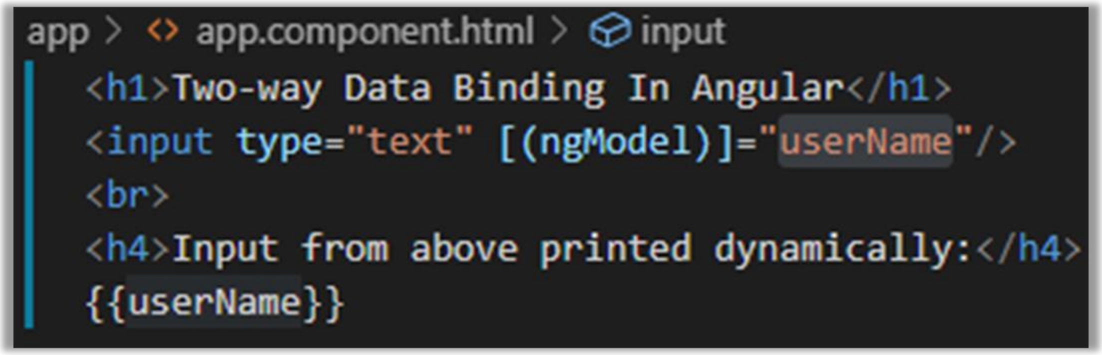


```
app > TS app.module.ts > ...
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import {FormsModule} from '@angular/forms';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

- Within the input tag in app.component.html file, add the directive ngModel and surround it with banana brackets, i.e. [( )]
- The square brackets are for property binding and parenthesis is for event binding
- To the ngModel directive assign the username property which is initialized to NULL in app.component.ts file



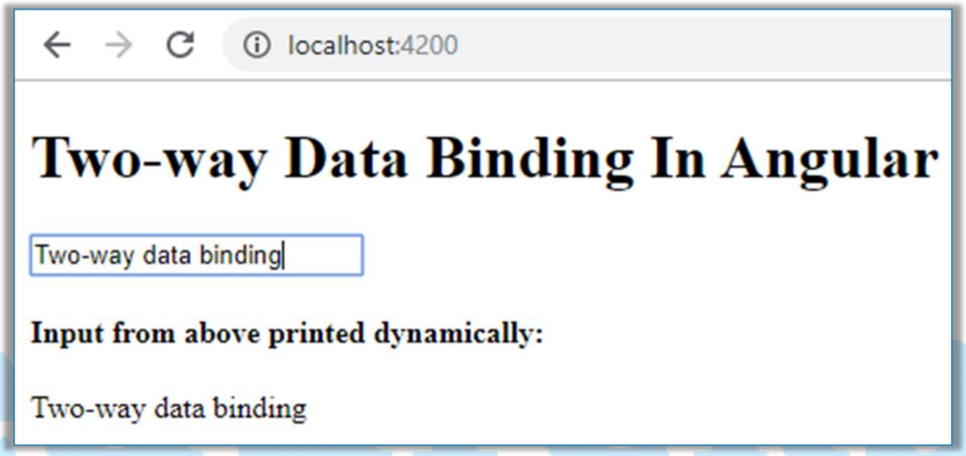
```
p > TS app.component.ts > ...
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {
  public userName="";
}
```



app > <> app.component.html > input

```
<h1>Two-way Data Binding In Angular</h1>
<input type="text" [(ngModel)]="userName"/>
<br>
<h4>Input from above printed dynamically:</h4>
{{userName}}
```

Step 2 – Whatever input is typed in the input box, same is reflected on the html page



localhost:4200

Two-way Data Binding In Angular

Two-way data binding

Input from above printed dynamically:

Two-way data binding