**Databinding**

1. Points 1 to 8 Contains Intro on Databinding
2. So Far we learnt following in angular:
3. How Angular starts
4. b) How to use Components
5. Thus far, we have done output of only static content.
6. To do Dynamic Content Output we use – DataBinding:
7. **What is Databinding:**

* Databinding helps communication between Typescript[*TS file*] code of your component[Your business logic] and the Template[*HTML file*]
* We may have some result in our typescript code -> which is fetched from the server, which needs to be displayed to the user.

1. **Different ways of communication:**

**String Interpolation:**

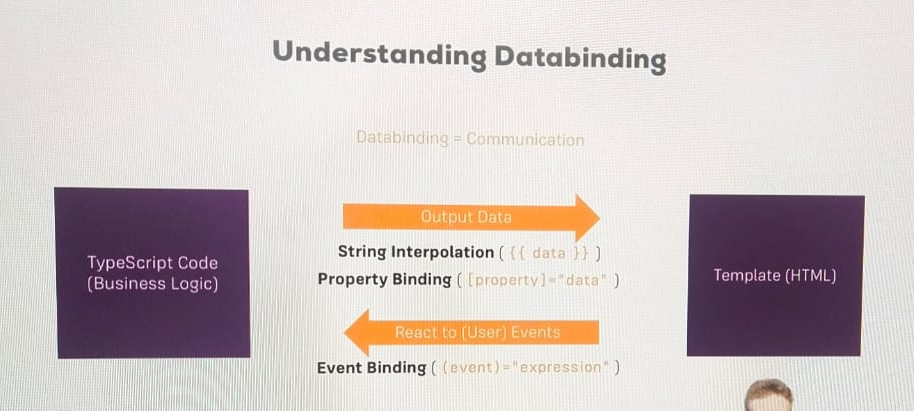
* If we want to output data from Typescript code to HTML code in the template – we can use String interpolation
* It is syntax with double curly braces: {{ data }}

**Property Binding:**

* It is syntax with square brackets eg: [property]=”data”

**Event Binding**

* When use clicks a button on html template, we may want to trigger something in our Typescript code
* So now we need the other direction, We can react to user events with event binding
* Eg: we can bind click event to execute some code whenever it occurs

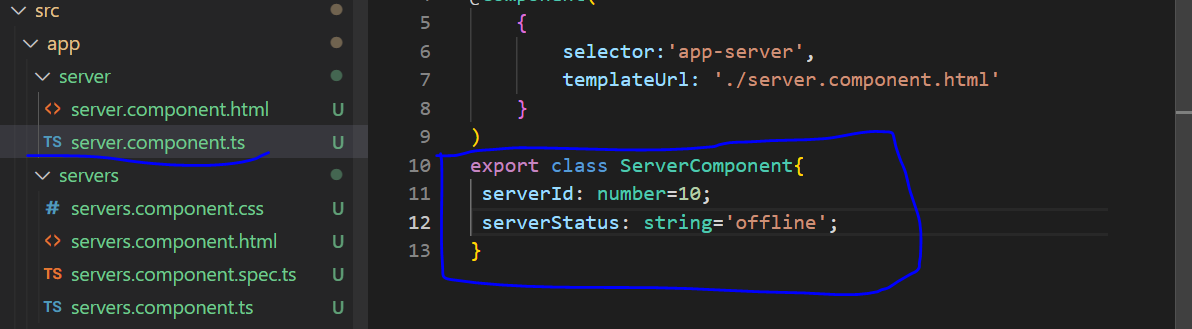
****

**Two Way – databinding**

* We combine both directions
* Two way data binding – where we are able to react to events and output something at the same time.

1. **String Interpolation**

* To print dynamic items in html :
* Lets have serverId hardcoded in Typescript file and print the same in html file



* Now to use the same in html file, we use String interpolation syntax: {{serverId}}.
* In between the brackets we can use typescript expression also

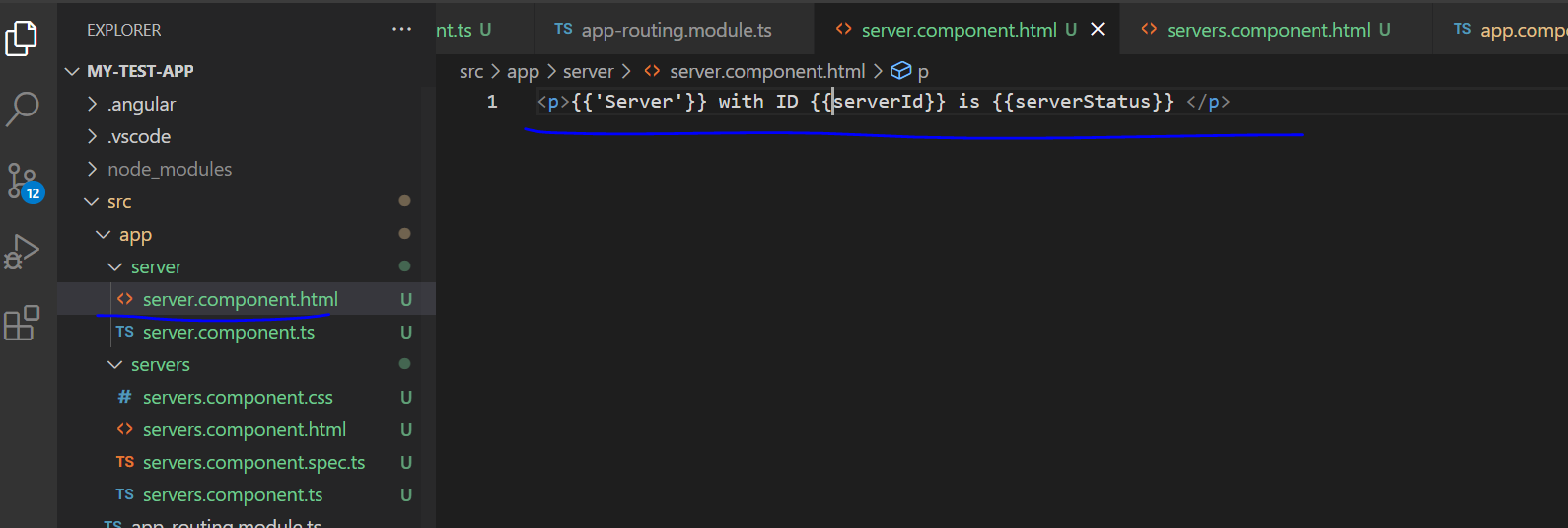
**Conditions for String interpolation**

1. Any expression should be resolved to a string in the end -> This is the only condition for String interpolation syntax

* So whatever we have in between the curly braces, it some how has to return a String. We can also call a method within the curly braces.

1. Can’t write multiline expressions here. But we can use ternary expression

* One more interesting thing here in the example here is: serverId is not a string and it is a number
* Here we can see server is given within single quotes since they are hardcoded and serverId and serverStatus are given directly for databinding



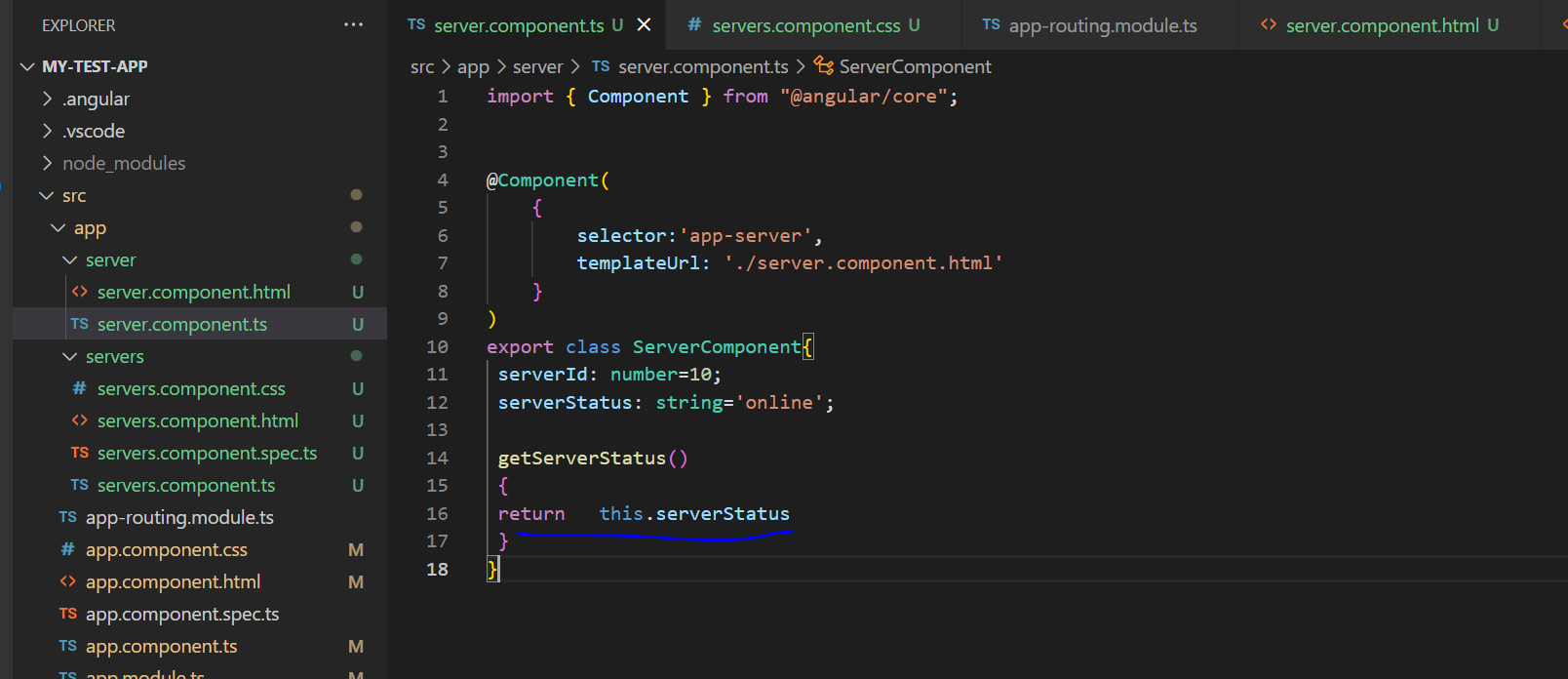
* In the browser we can see the below output

Graphical user interface, text, application, email

Description automatically generated

**Calling a method using Typescript:**

* Also we can display the server status by defining a method in typescript



* We can call this method using the below line in html

Graphical user interface, text

Description automatically generated

1. **Property Binding:**

* Lets add a button and add below code for the same.

Text

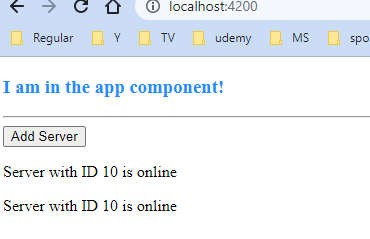
Description automatically generated

* To make it reflected in UI, lets add the below changes to take template url instead of inline template in servers.component.ts file

Text

Description automatically generated

* Now if we check in browser we can get the following output

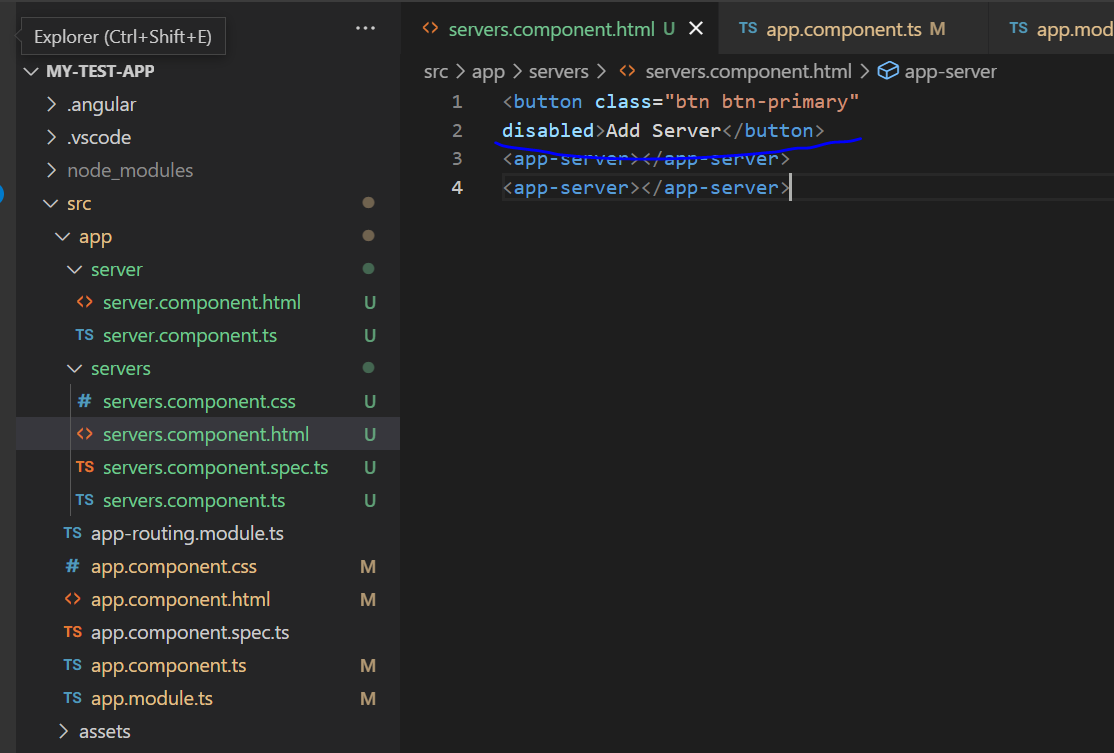


* Next I will add a new property in typescript here

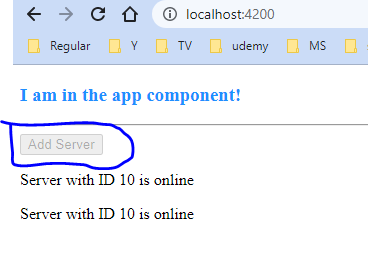
Text

Description automatically generated

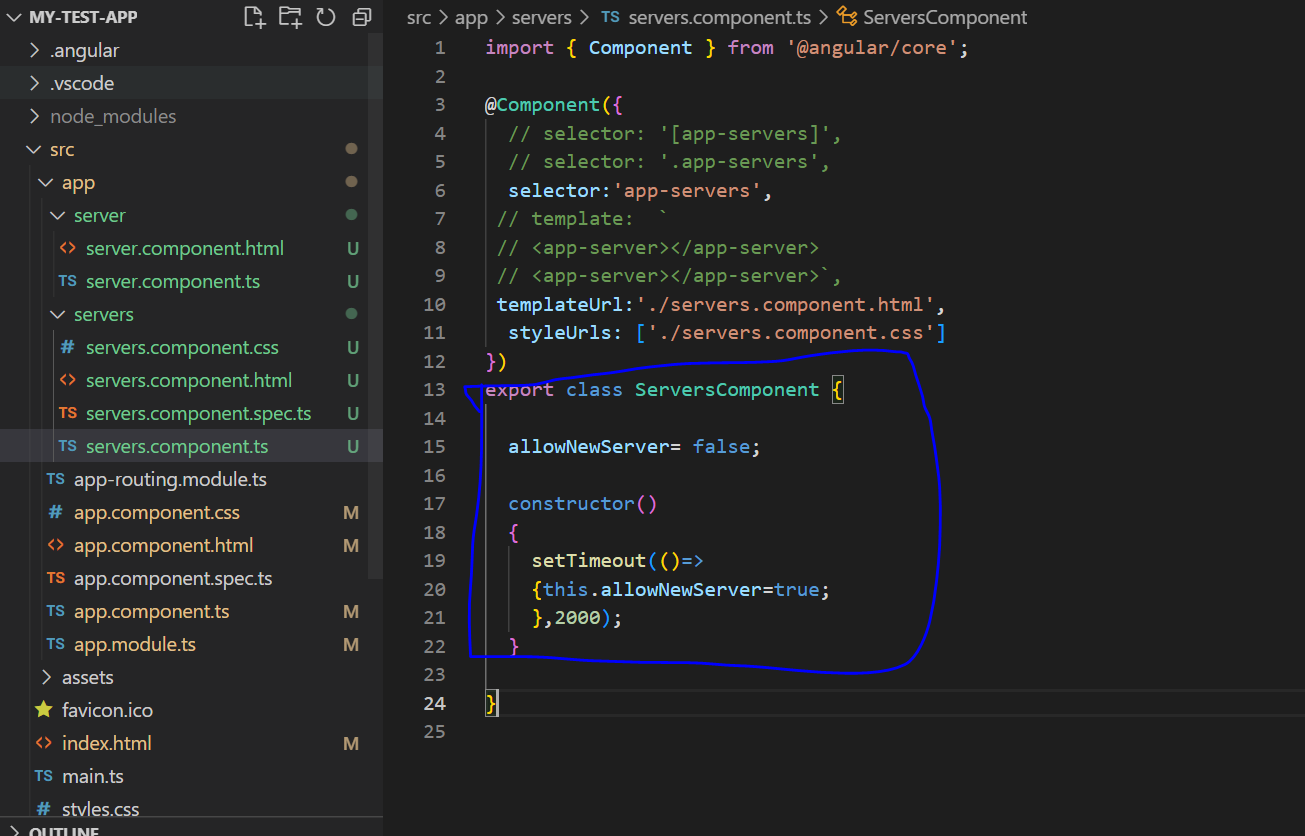
* Now in the html file , I want to disabled the button, for which there is disabled attribute which we can add.



* Now after making below change, in UI the button gets disabled



* No in ServerComponent constructor, lets set timout to 2000 ms.
* Now we will use ES6 array function:



setTimeout()=>{}

* It is almost similar to :

Function(){}

* But with some difference of handling the this keyword.
* In Es6 we pass arguments and right side we have function body. It is similar to lambda function of java
* To make disabled variable of servers.component.html binded, we can enclose it inside square brackets. Here square brackets indicate angular, that we are using property binding meaning we want to dynamically bind some property and disabled the html attribute
* In the end, normal html only sets a specific property on the underlying dom element.
* We know that each html element we use is parsed by the browser and kind of translated into an element on document object model.
* Here the element is button.here using allowNewServer property we bind this property to this button, based on a Boolean expression

Text

Description automatically generated

* Now we can see that as a result of our changes, add server button gets disabled, while loading and then gets enabled after 2 seconds