1. **Event Binding:**

* Eg: If we have a button here and after two seconds this button is enabled
* It will be nice if we could do something, once we click the button.
* Lets add a new replication of our server component here.
* For now I want to output something. So I will remove the below line:

Text

Description automatically generated

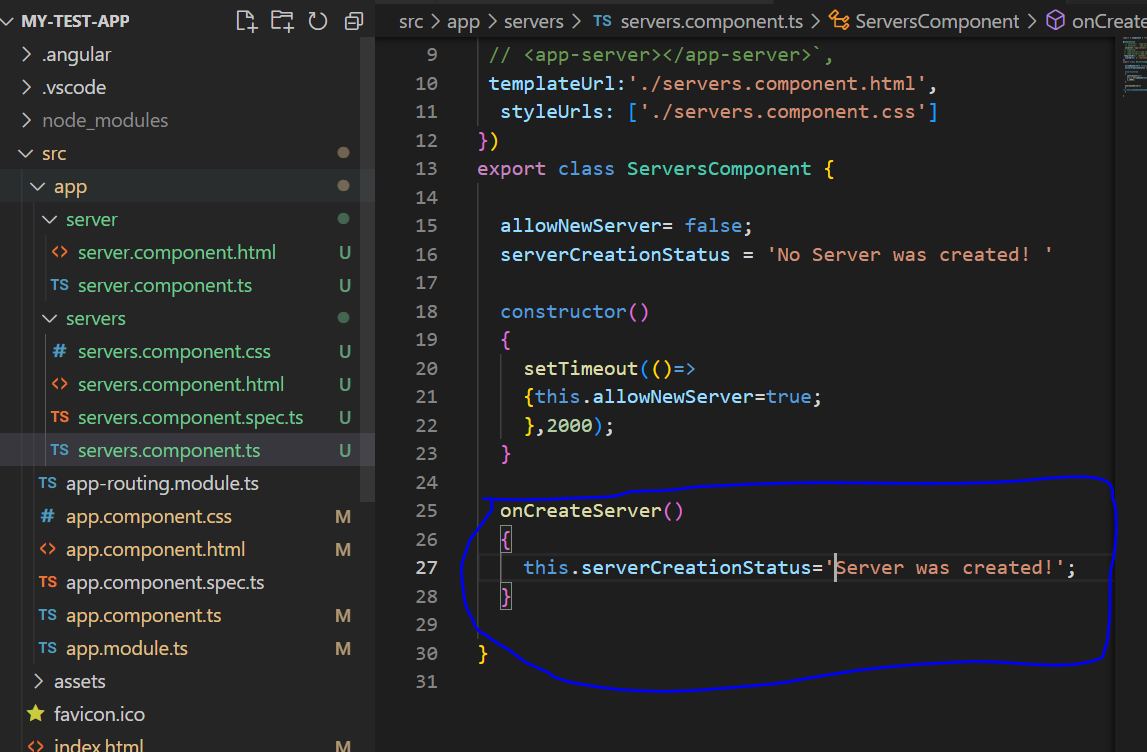
* Instead I will output the value of a property here. I will create the property: serverCreationStatus



* Now lets output the same in html file using String interpolation

**Achieving Event Binding**

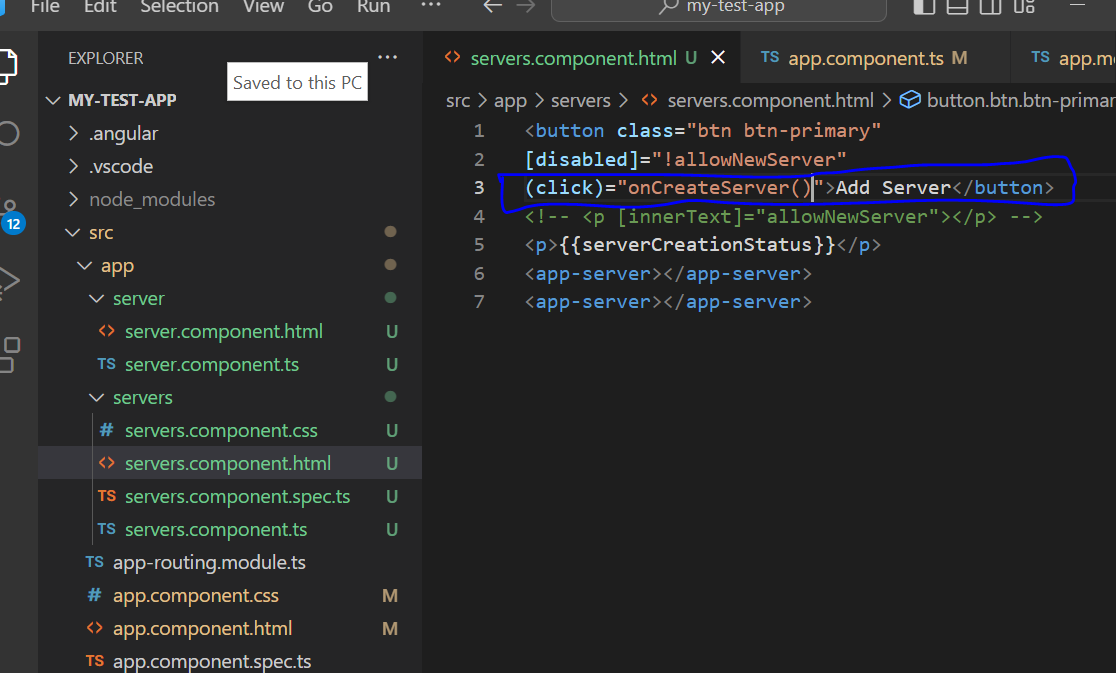
* But our motto now is not to output the string, but to listen to events. We will add another method in ServersComponent : onCreateServer()



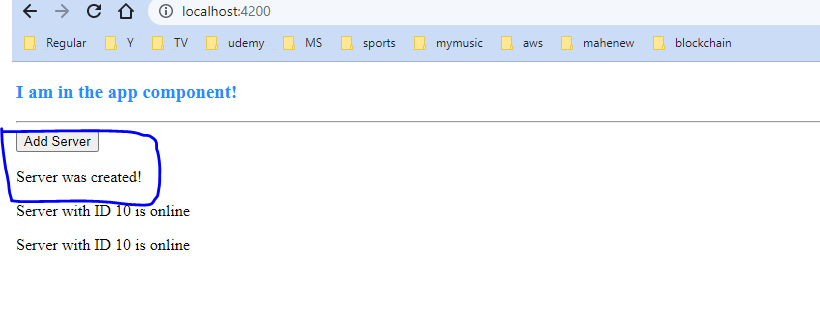
* Now is where eventBinding comes into play. We want to listen to click event here in servers.component.html ‘s button element.
* We have onClick attribute on the html element, where we can execute some java script code. But we wont use it here with Angular. Instead angular offers us Event binding.
* Event binding uses parenthesis, while property binding used square brackets.
* For events parenthesis are signals that we are going to use event binding.
* Name of the event should be specified in parenthesis eg: click
* We can bind to all the events made available by the html attribute, we are placing this on
* Hence if you have on click, on mouse enter, you could bind to click mouse enter, without the “on” at the beginning:

Eg: (click) = “”>Add Server</button>

* Now between quotation marks, we put the code which we want to execute when the event occurs.



* In most scenarios, we will call a method here. So we will point to the method which we want to execute: In our case it is “onCreateServer()”
* So onCreateServer method will get executed whenever we click the button
* Thus we have binded the click event and we execute the onCreateServer() when it occurs
* We can observe the same in browser also.

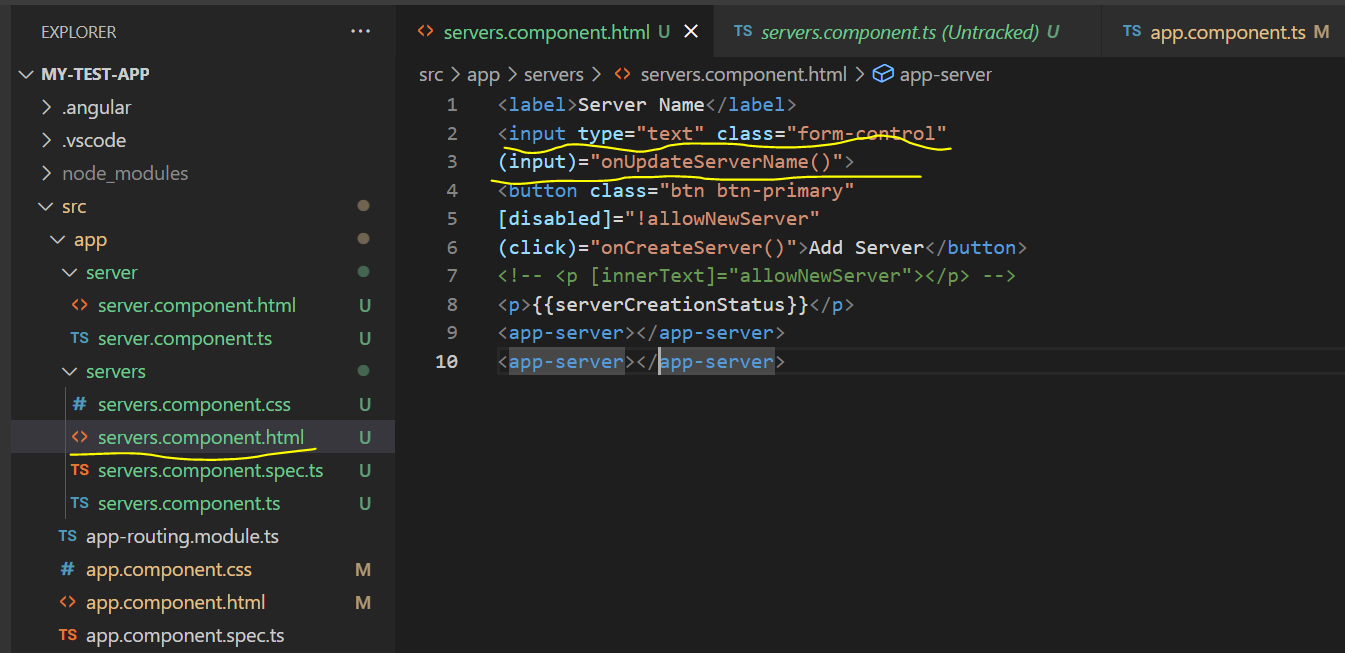


1. **Bindable Properties and Events**

* How do you know to which Properties or Events of HTML Elements you may bind? You can basically bind to all Properties and Events - a good idea is to console.log()  the element you're interested in to see which properties and events it offers.
* Important: For events, you don't bind to onclick but only to click (=> (click)).
* The MDN (Mozilla Developer Network) offers nice lists of all properties and events of the element you're interested in. Googling for YOUR\_ELEMENT properties  or YOUR\_ELEMENT events  should yield nice results.

1. **Passing and Using Data with Event Binding:**

* We need to know one important thing in Event Binding
* Lets say: we have a label and an input before the button in servers.component.html page



* So we add label and input element. The class is form-control for input
* Here we are listening to the input event . This is a normal DOM event provided by the input element. Here we listen to the onUpdateServerName(), because it will be fired on every key stroke.
* <input type="text" class="form-control"
* (input)="onUpdateServerName()">
* Now we add this method to the component in servers.component.ts file

**Dollar Sign event($event):Gives access to event data**

* Also I want to output whatever value what I entered . That can be done by adding $event to the function name passed in html file

Eg:

(input)="onUpdateServerName($event)">

* $event is a reserved variable name , we can use in the template while using event binding
* For the event: "onCreateServer()", dollar sign event ($event) will simply be the data emitted with that event.

**Some default events:**

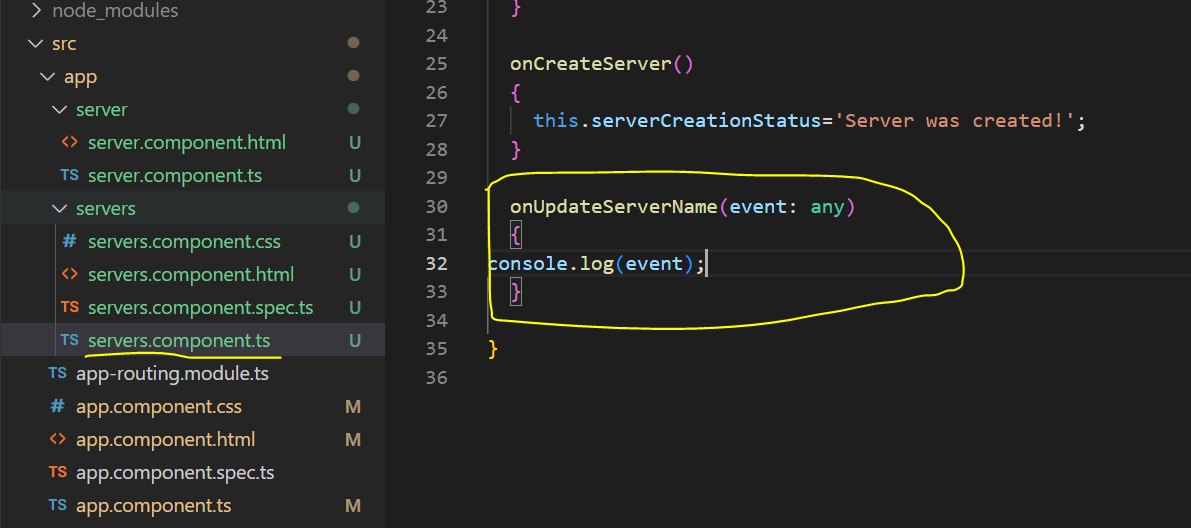
* Input and click are default events provided by the dom. They give us some data when they are fired.
* The click events gives us the object: eg it holds the coordinates where we clicked.
* Input event gives us some information about the event

**Capturing data from Event:**

* We can capture the data returned by the event, using the dollar sign event($event) passed as an argument to the method we are calling or used anywhere between these quotation marks, in the code we are executing

**onUpdateServerName:**

* Here we pass the $event to onUpdateServerName. Here we know we will receive this event and this will be of type any for now
* Now lets simply log the event to the console, so that we can see it in the debug log

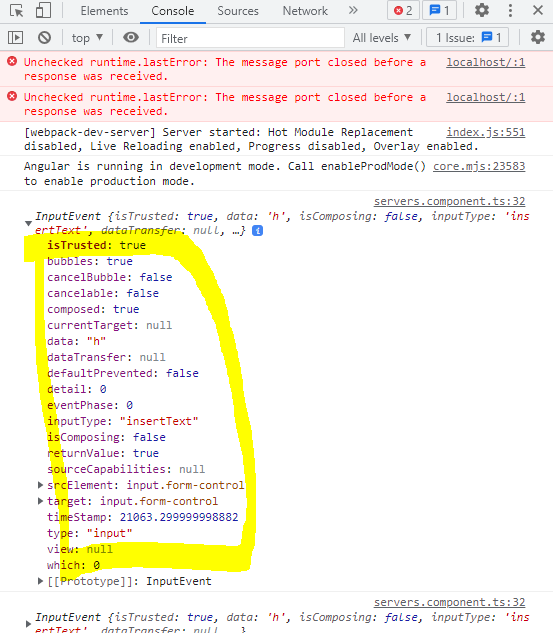


* Now if we check the same in browser, we can see in console that event data getting printed, for every entry in input

Graphical user interface, text, application

Description automatically generated

* Now if we check the console output, we can see there is target. Target is simply the html element on which this event occurred.

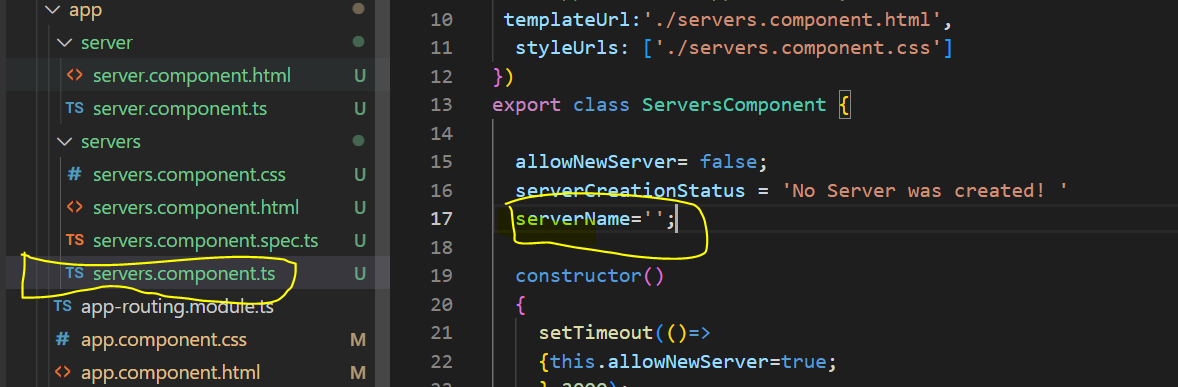


* On the target if we scroll down we can see the value, which is the actual value entered by the user.

Graphical user interface, text, application

Description automatically generated

* In our typescript code(servers.component.ts), we can set a server name property which is an empty string by default.



* In onUpdateServerName() , we can update server name from event.target.value

Eg:

  onUpdateServerName(event: Event)

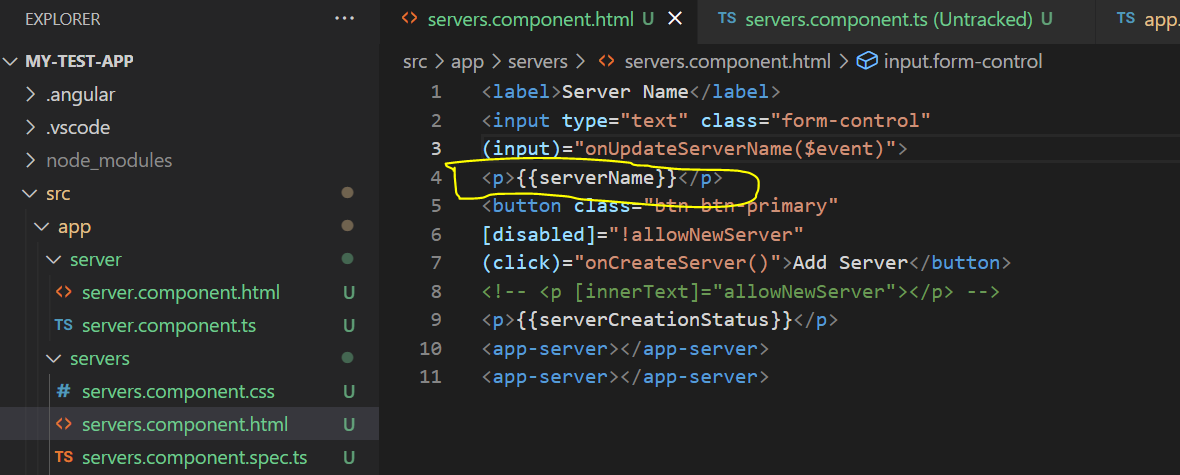
  {

 this.serverName = (<HTMLInputElement>event.target).value;

  }

* Here we need to cast it to HTMLInputElement to get the value , this is needed to inform typescript : that we know that the type of the HTML element of this event will be an HTML input element.
* Now in the HTML file(servers.component.html) file, we can output the value of the servername entered in input , using String interpolation

Eg:



* Since the serverName value is updated in onUpdateServerName, we will get the input value displayed below: