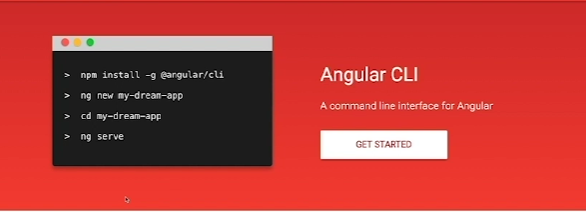


 Angular is a framework which allows you to create reactive single page applications.

A diagram of angular version

Description automatically generated

Installation and creating first app:

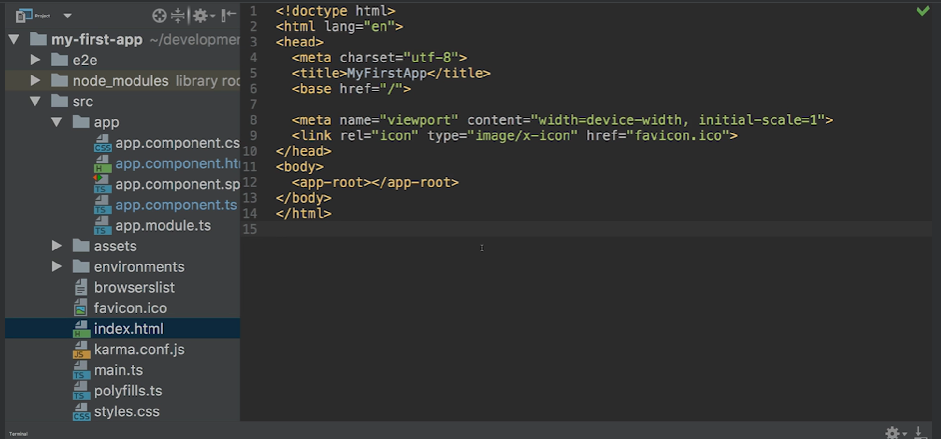


Command to run angular app: ng serve

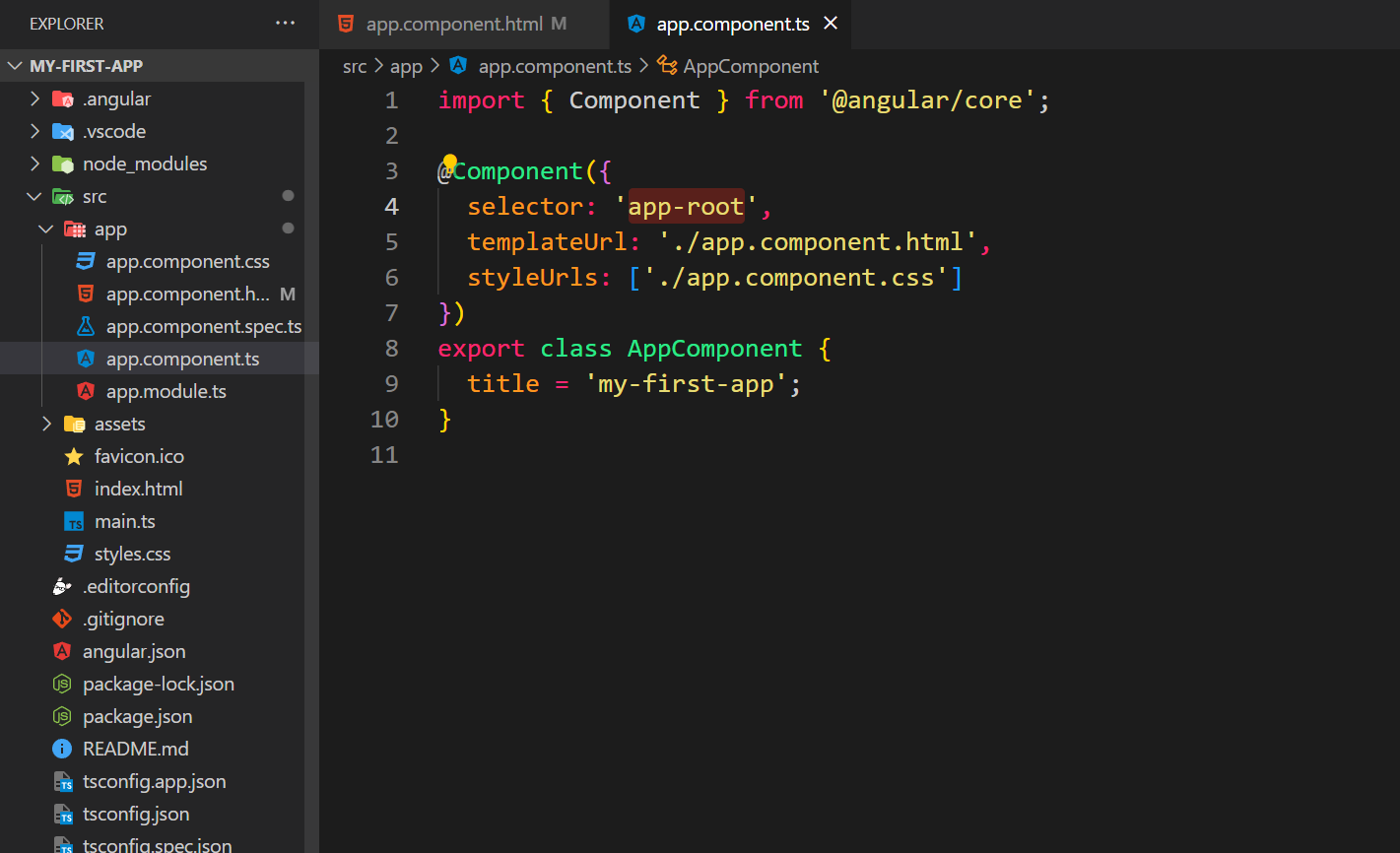
Go to browser and write in address bar: <http://localhost:4200/>

Working on Angular:

Structure:



First load the index.html which has <app-root> tag. This represent the app component in app folder.



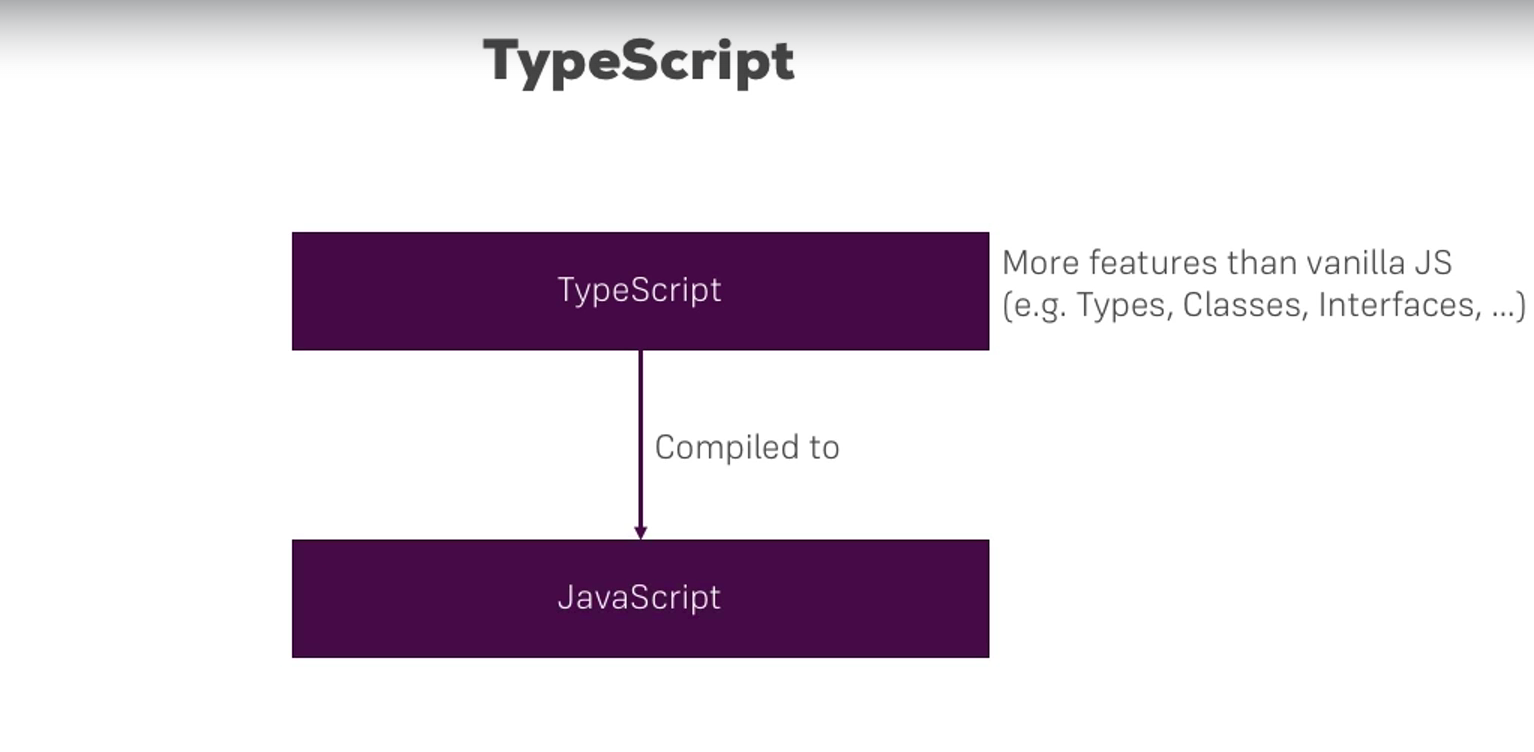
So, app-root is a selector of appcomponent and this is mentioned in the index.html

So, run index.html first then it will call the component whose selector is app-root.

Topics:



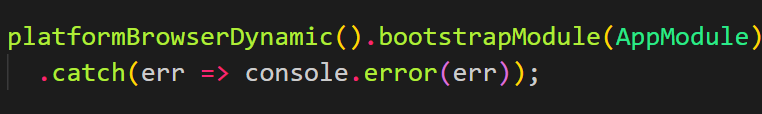
TypeScript:



Angular Flow:

Main.ts 🡪 app.module.ts(has the bootstrap array) 🡪 Index.html 🡪 root component or app component 🡪

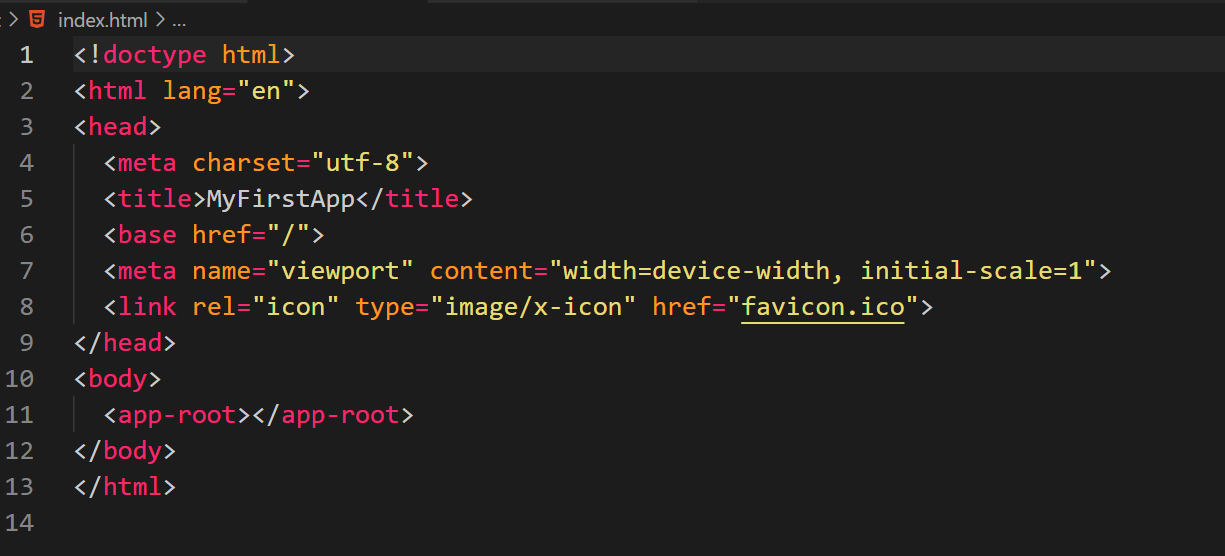
Main.ts has this:



App.module.ts has below:

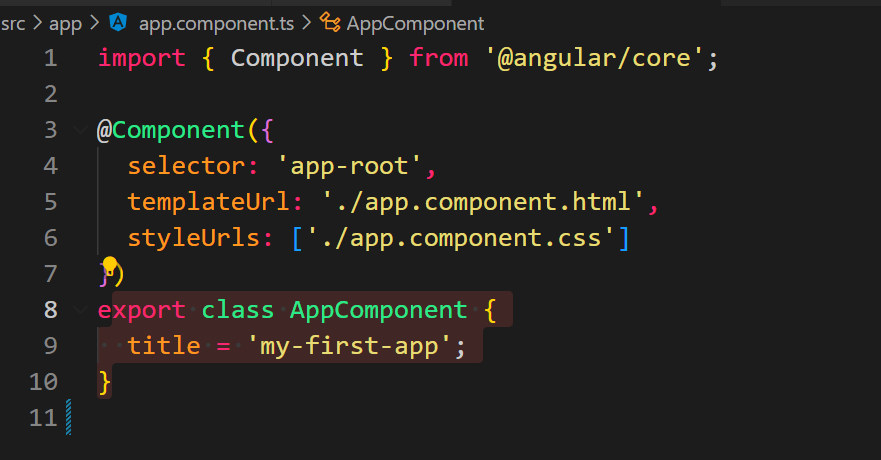


Bootstrap array tells us which component to load at the start.

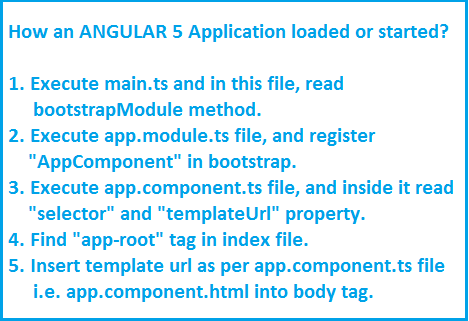
Index.html has app-root selector:

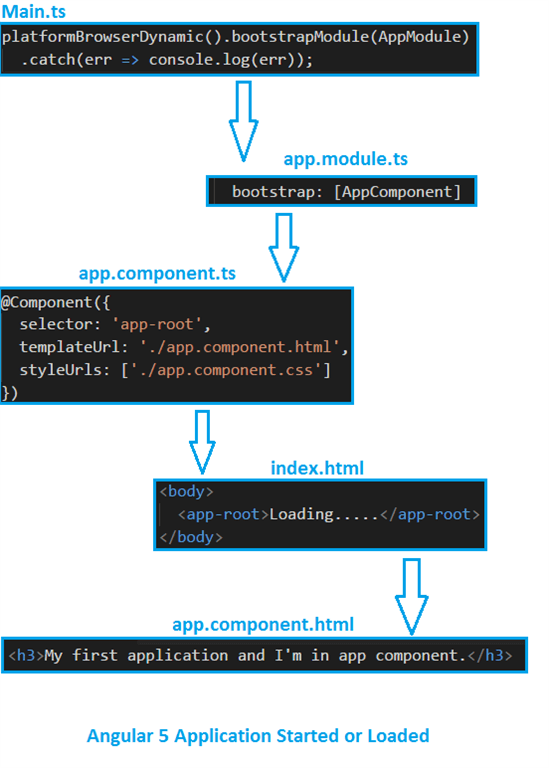
This selector selects the component which has the selector as app-root.

App.component.ts has:



Start-up:



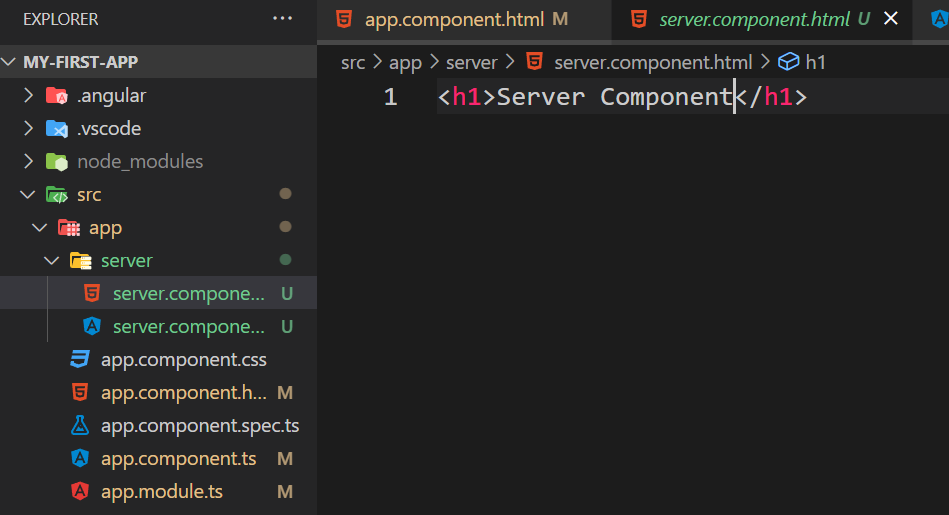


Components:

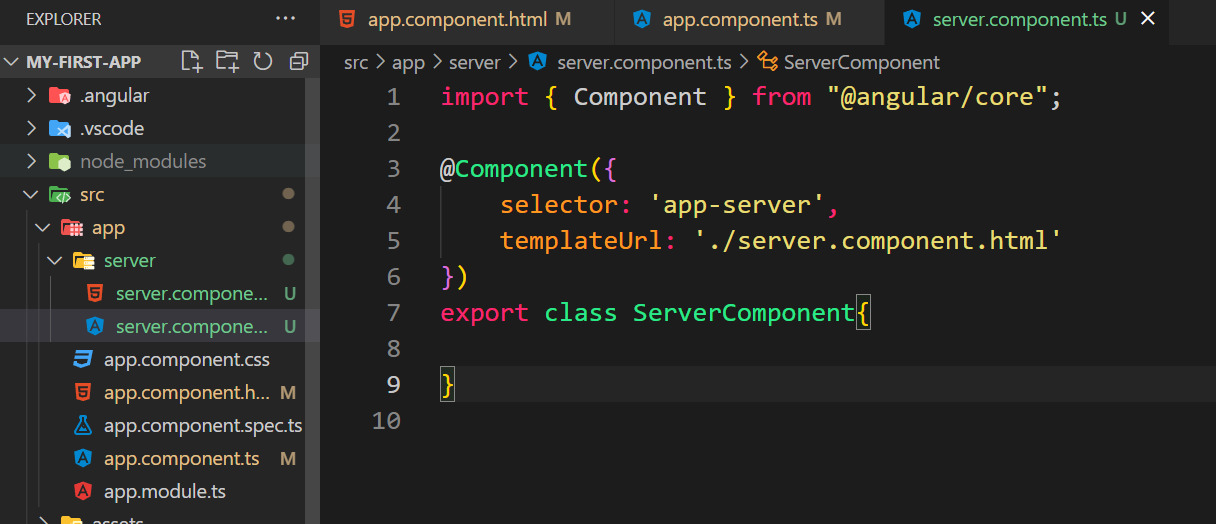
AppComponent is the start component of angular. When you write your component their selectors are added to the AppComponent.

How to create component:

1. Create a folder with the name of the component inside the app folder.
2. Create <name>.component.ts file inside the created folder.
3. Create <name>.component.html file, you will write the html code of your component inside this file.



1. In the <name>.component.ts file write this code:



Import the component.

Write @Component directive and pass selector and templateUrl inside that.

1. Register your component in app.module.ts in declarations also import it at the top of this file. See below:



Now you can write the selector in app.component.html and your component can be show.

These are too many steps. We can create angular components directly using CLI.

We can use below commands to create angular components:

ng generate component <component-name>

or

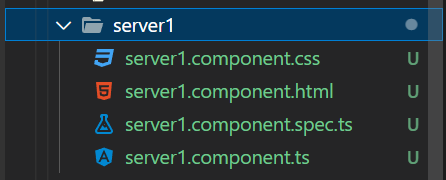
ng g c <component-name>

Angular will create a folder with your component name inside the app folder.

This will have four files:

HTML, CSS, TS and a spec file which is used for testing.

Here we have created a component called server1. See the files created:



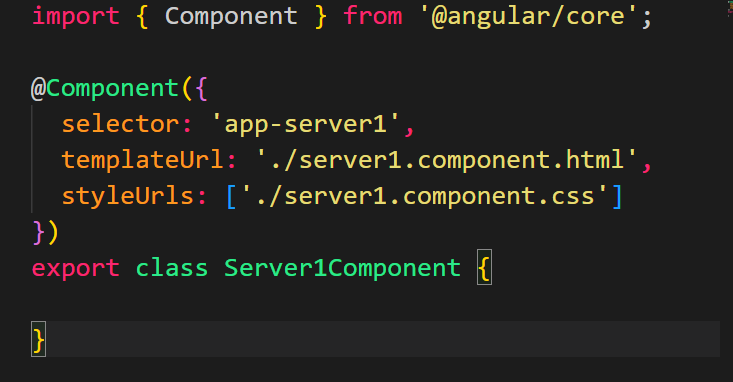
Once the component is created you can reuse this component as many times as you want by simply writing the selector of your component.

.html file is used to write the html for the component.

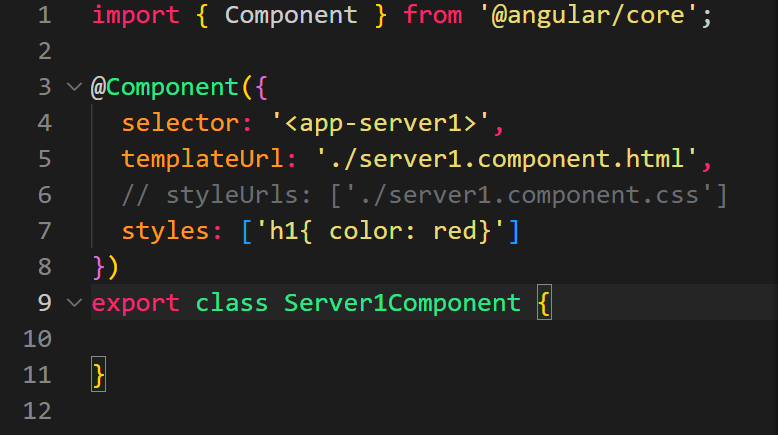
.css file is used to write the style/css for the component.

.ts file is used to write script/js for the component.

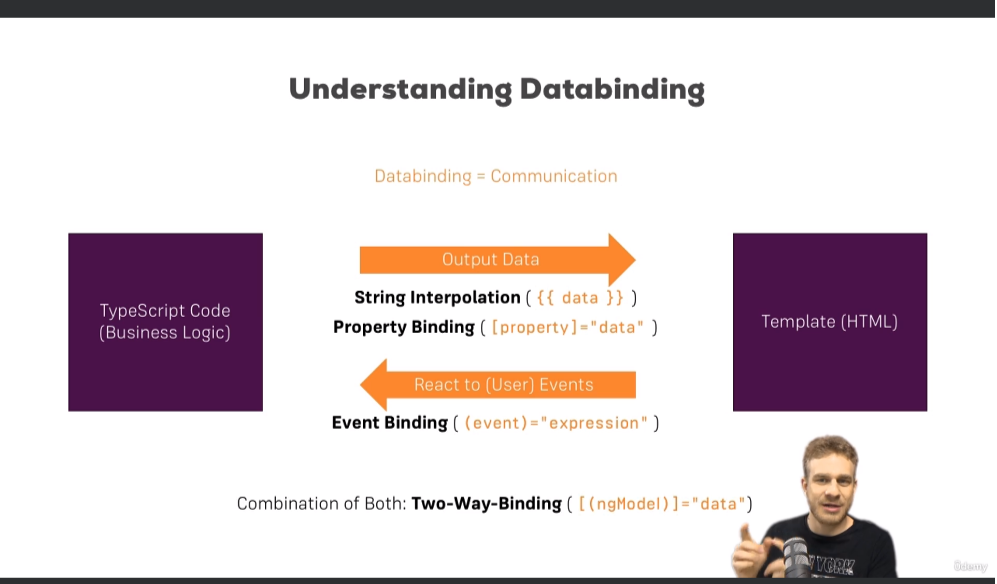
See the ts file:



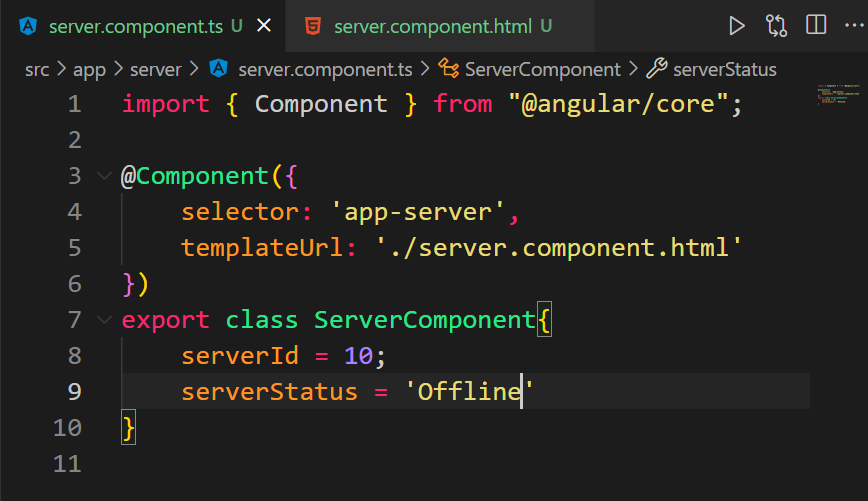
StyleUrls will tell you this CSS files which are used in this component. Notice that it is an array. You can also write your style as shown below:

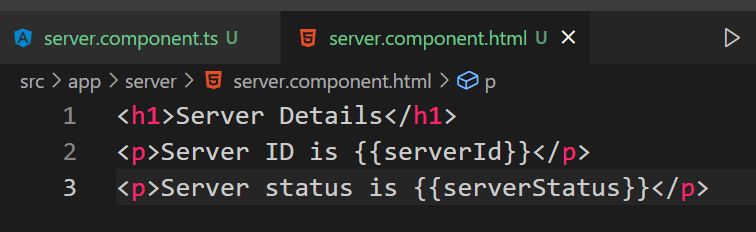


Data Binding:



String Interpolation:



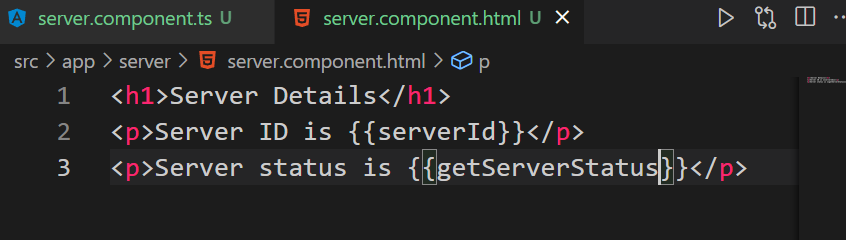


Put your typescript variable inside {{}} in the template html.

Also, remember that string interpolation has to return a string. So, whatever is there inside {{}} will ultimately be converted to a string and then shown in template.

You can also call a typescript function in string interpolation which returns a string.

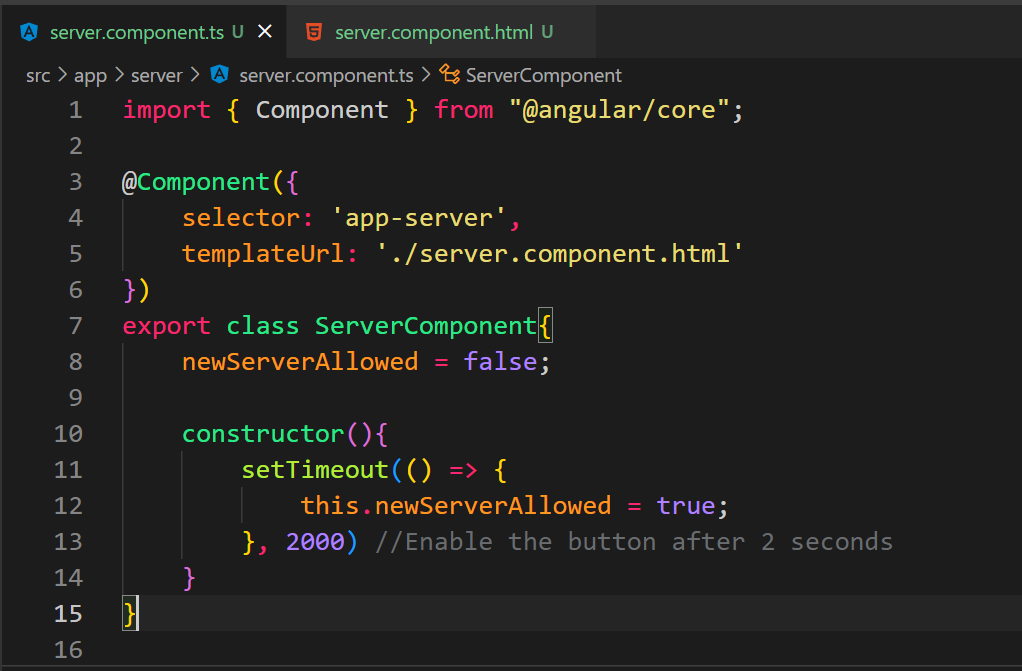
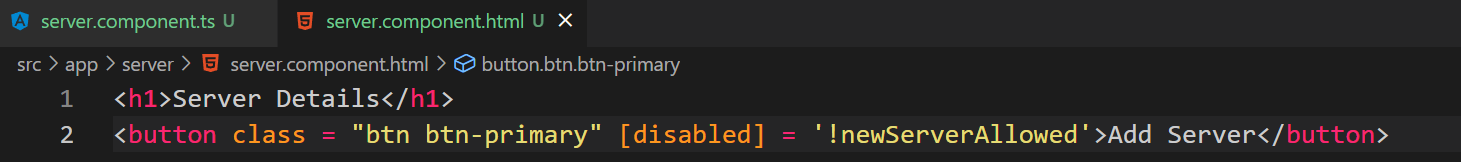




Property Binding:

It is used to set the property of html element from typescript variables.

You can bind a property/attribute of html element to the TS variable.

Button’s disable property is bound to the newServerAllowed boolean of TS.

In the constructor, newServerAllowed is set to true after 2 seconds.

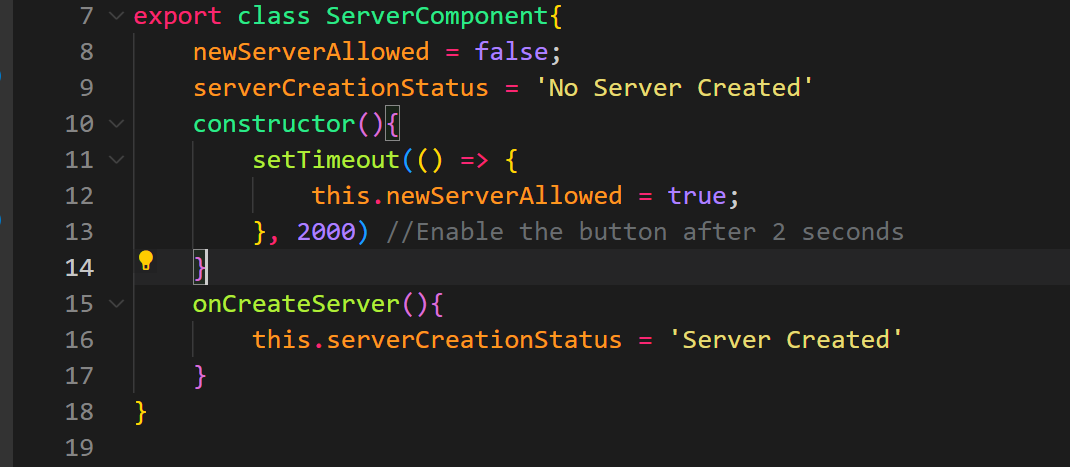
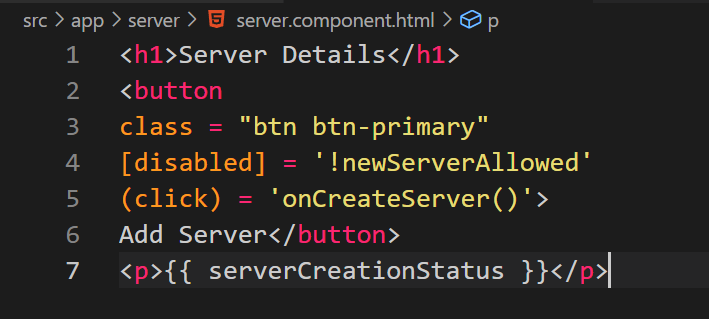
So, the button is enabled after two seconds.

if you want to output something in your template print some text to it, use string interpolation.

If you want to change some property, be that of an HTML element or as you later learn off a directive or a component, typically use property binding.

Event Binding:

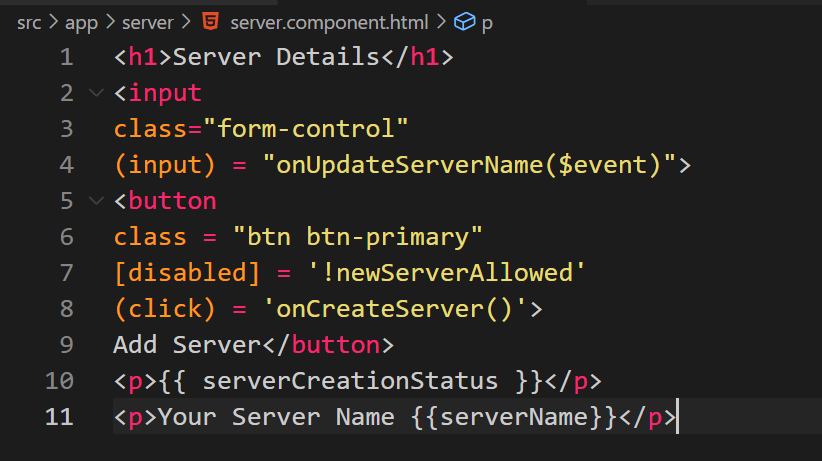
It is used to bound event of an html element to a TS function. Eg binding a click event of a button to a TS function.

Here we have bound click event of Add server button to onCreateServer() TS function.

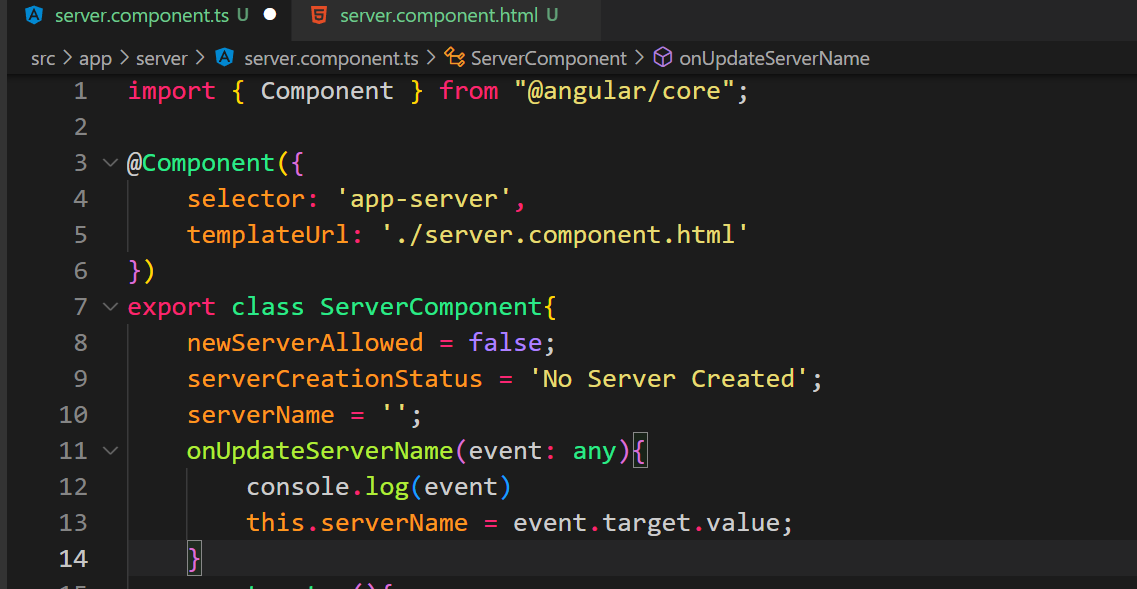
This TS function changes the TS string serverCreationStatus which is interpolated in the p tags in the next line.

So, as you click the button No Server Created string is changed to Server Created on the screen.



See the input tag, we have bound the input event of this tag to onUpdateServerName() function of TS. From here we are passing $event to this method.

$event will simply be the data emitted with that event. To see what data we got you can console.log the event in TS.



Important: For Two-Way-Binding (covered in the next lecture) to work, you need to enable the ngModel directive. This is done by adding the FormsModule to the imports[] array in the AppModule.

You then also need to add the import from @angular/forms in the app.module.ts file:

import { FormsModule } from '@angular/forms';

Two Way Binding:

It is used to send data from TS to template/html and template/html to TS.

So, data can go both ways. With two-way data binding,

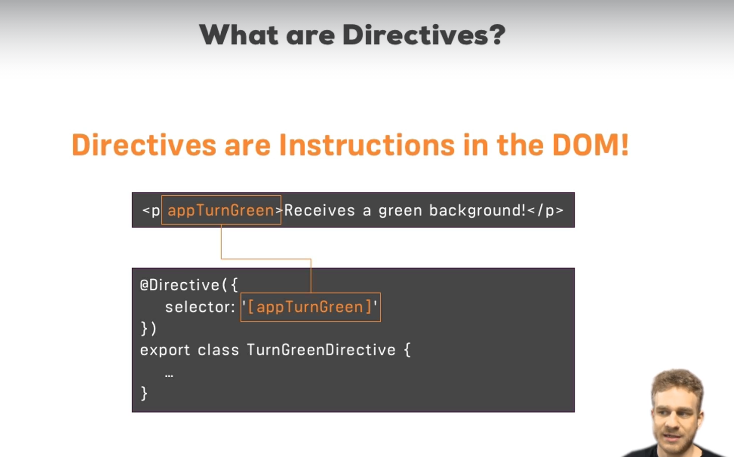
we combine property and event binding. With two-way data binding,

we combine property and event binding.



Here input is bound to TS variable serverName. If you change serverName anywhere in TS, this will change input. If you change the input which will change the TS variable.

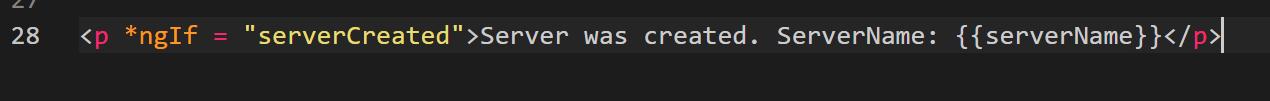
Directives:



ngIf: It is a structural directive.

It is used to add html element into the DOM conditionally.

If the condition given in the ngIf is satisfied then it the html element is added to the DOM, else it is not added.



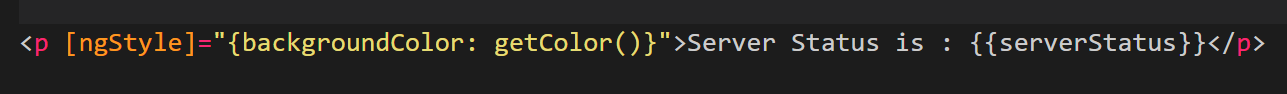
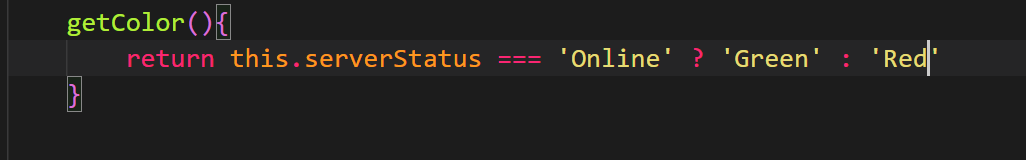
This p element is added to the DOM if serverCreated variable in TS is true.

Else it is not added. Remember to always add \* at the beginning of ngIf.

Also, you can put any condition in ngIf, whatever you put it must evaluate to true or false.

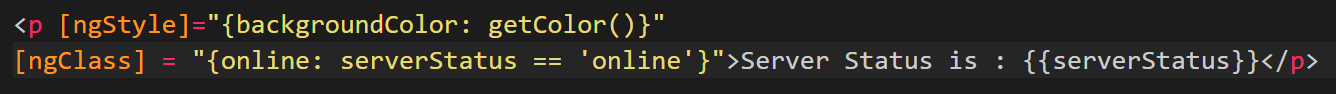
We can also add else condition with this.

ngStyle: it is an attribute directive.

Here we have used the directive ngStyle along with property binding.

ngClass: you can use this directive to apply a class conditionally.



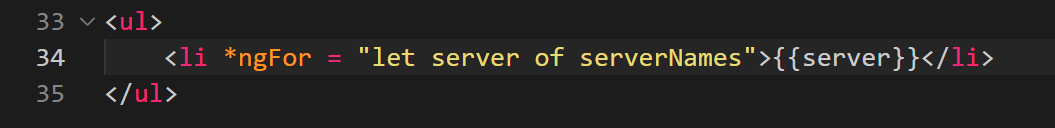
If serverStatus var of TS === ‘online’ then online class is applied else not.

ngFor:

It is very important. It is used to loop over a list. It can be used to show items of the list.

In the TS, we have:

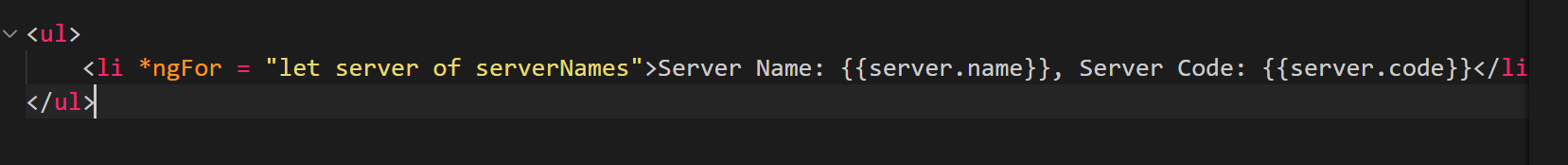




Output:



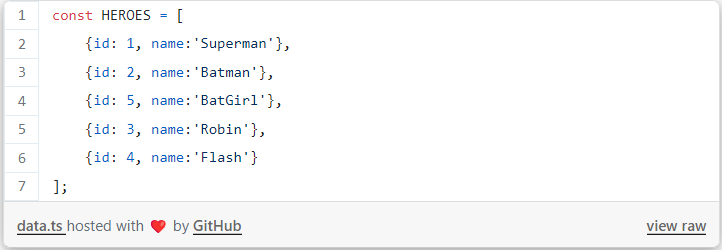
We can also loop over TS objects:

Output:



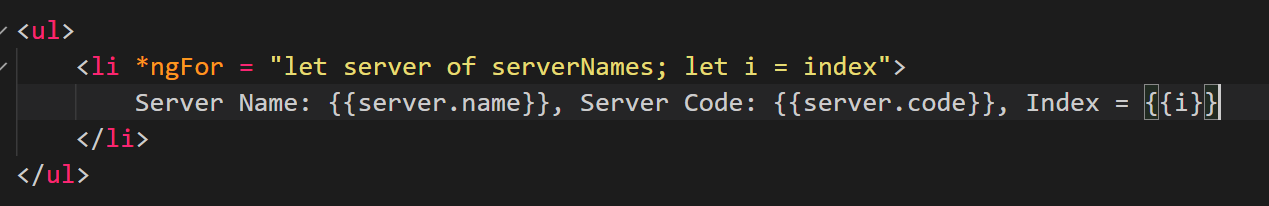
This can also be used to generate tables:

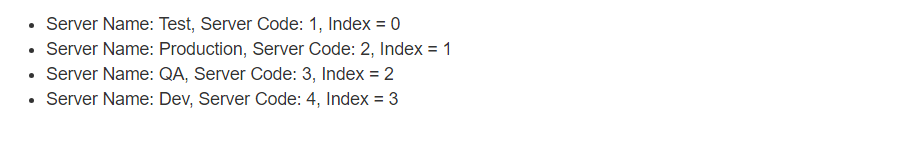




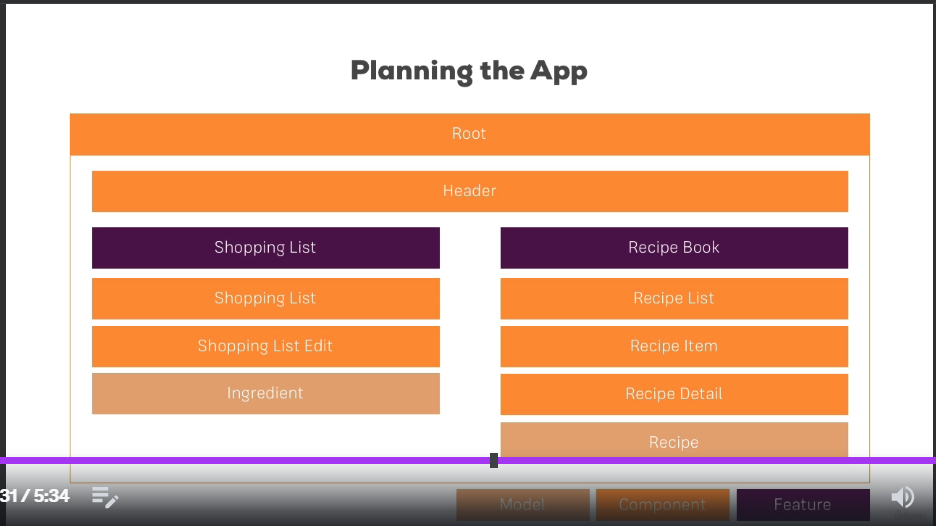
You can also get the index of the items showed in list.

Use:





Project:



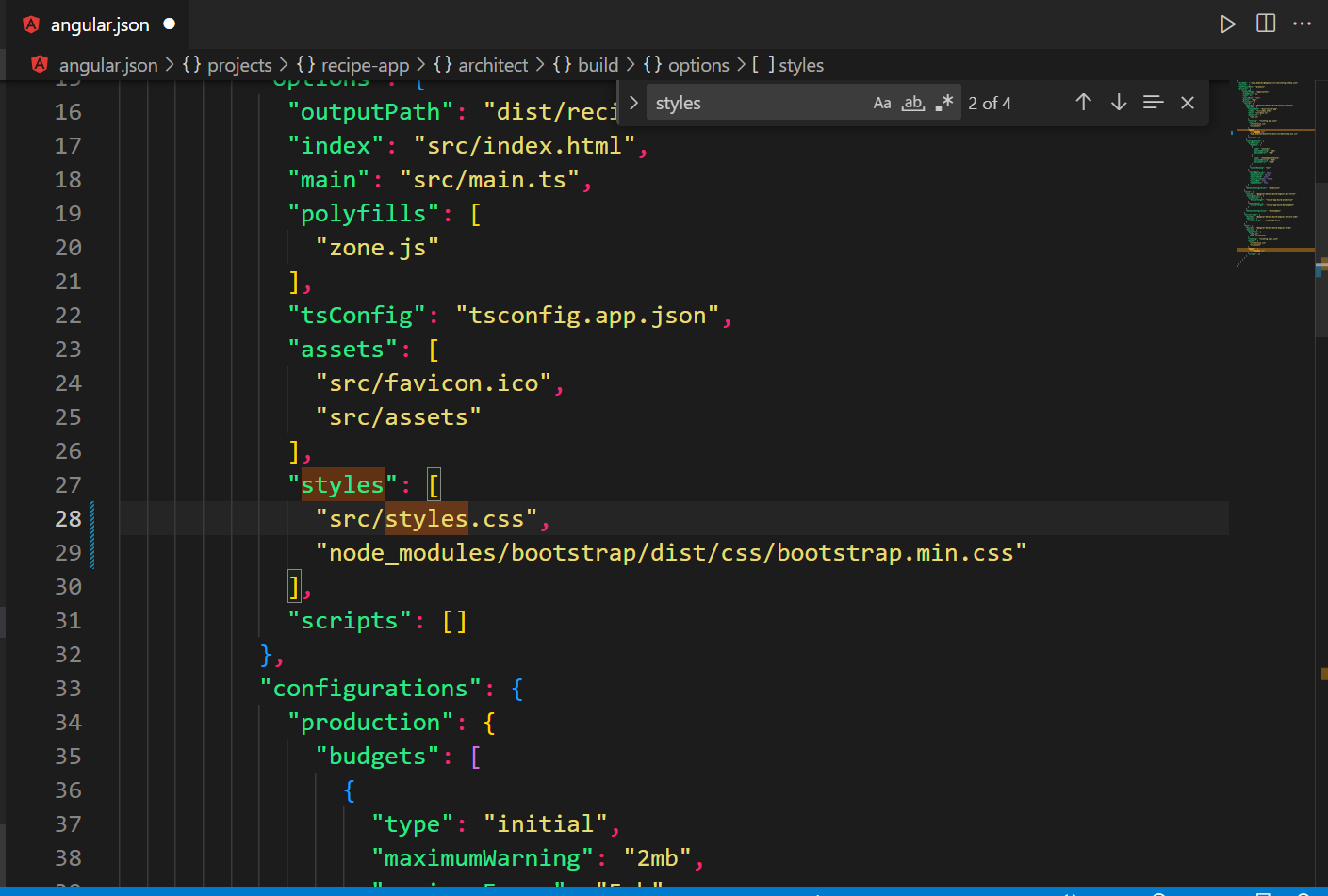
These are the Model (Data/Classes), Component, Features, we need in our recipe app.

Creating a new project: **ng** **new recipe-app** or **ng new recipe-app --no-strict** for strict mode.

Installing Bootstrap: First go to the folder of app just created:

**npm install --save bootstrap@3**

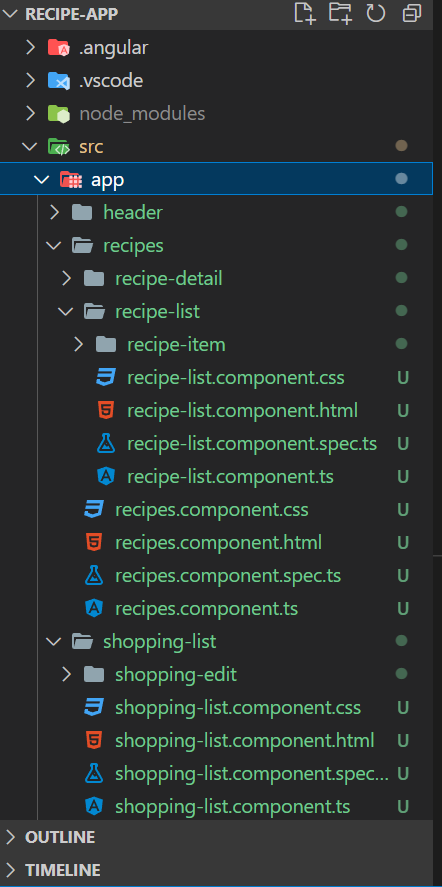
To add it to angular, go to angular.js file and add it in the styles array as:



"node\_modules/bootstrap/dist/css/bootstrap.min.css"

Remember if we add bootstrap in the angular.json file, this means that the style will be applied to the whole project.

When we create components, we must focus on the folder structure. Components which are related to each other should be created in same folder. Components should be nested based on how they work. See this folder structure we create for this recipe app:



Header is inside app folder.

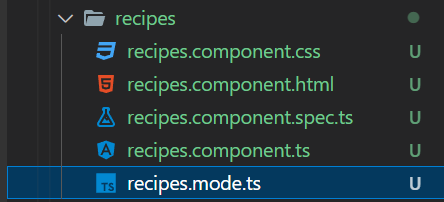
Then we have recipe folder in app folder. Recipe folder has recipe-detail and recipe-list folders. Recipe-list folder has recipe-item folder.

Remember all these folders are components.

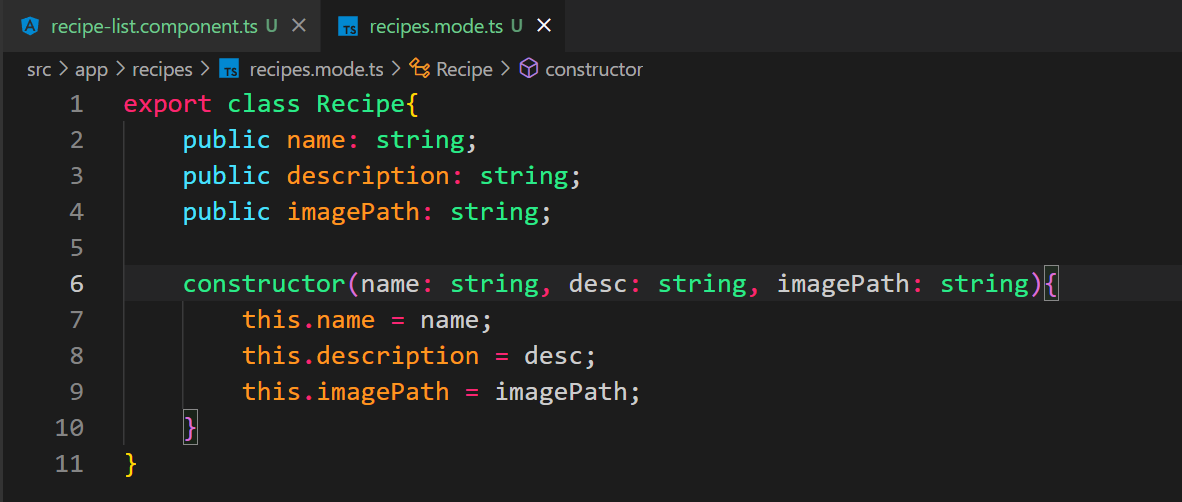
Here we are going to show recipes. Recipe will be a class and it will have some properties. So, we will create a model for this. A model is simple a JS/TS class which can be used to create objects whenever we need.

For this we create a new file inside the component folder. As:

For recipe model we create as:



The class looks like this:



Recipe will have three things: name, description and imagePath.

We also create a constructor, so we can easily create objects of this class.

Once model is created, we will have to add it in components TS file whichever component wants to use this model.

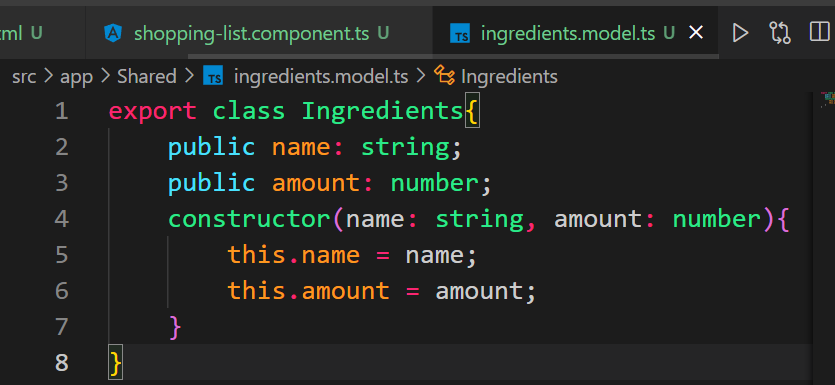
First step is to import this class in the component.

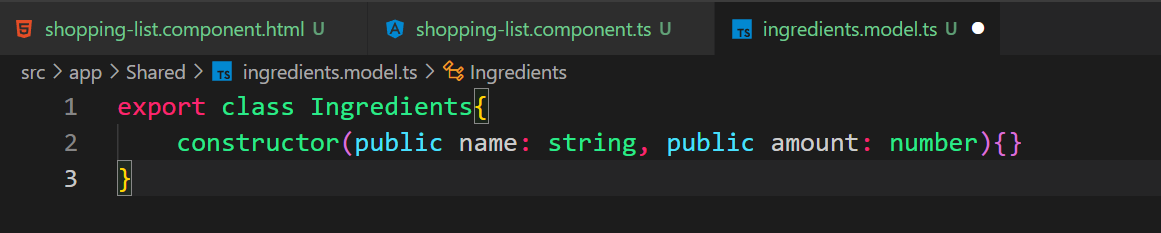
Second is to create an array or object of this class with values(for constructor).



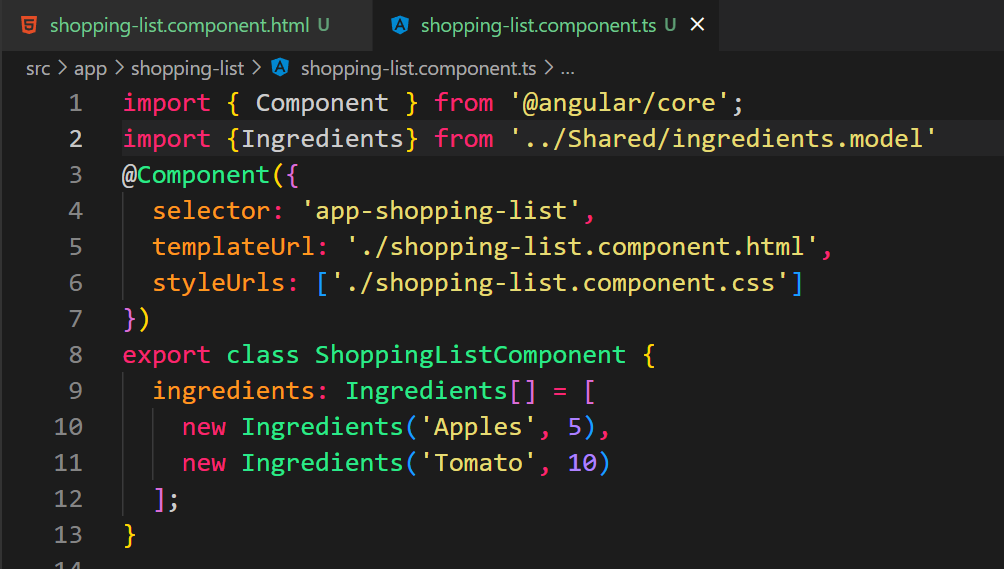
You can put ngfor on any element which you want to repeat.

See this model TS class for ingredients for our recipe:



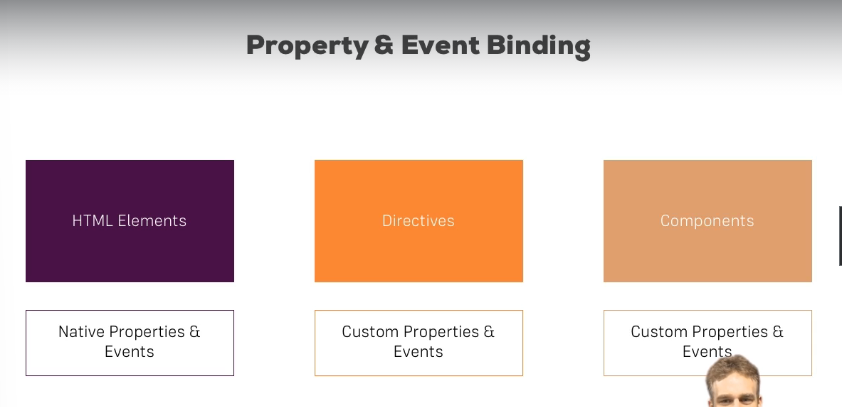
Instead of declaring properties separately and then using constructor, we can use constructor directly to declare and set values to the properties: 

Using ingredients in shopping-list component:



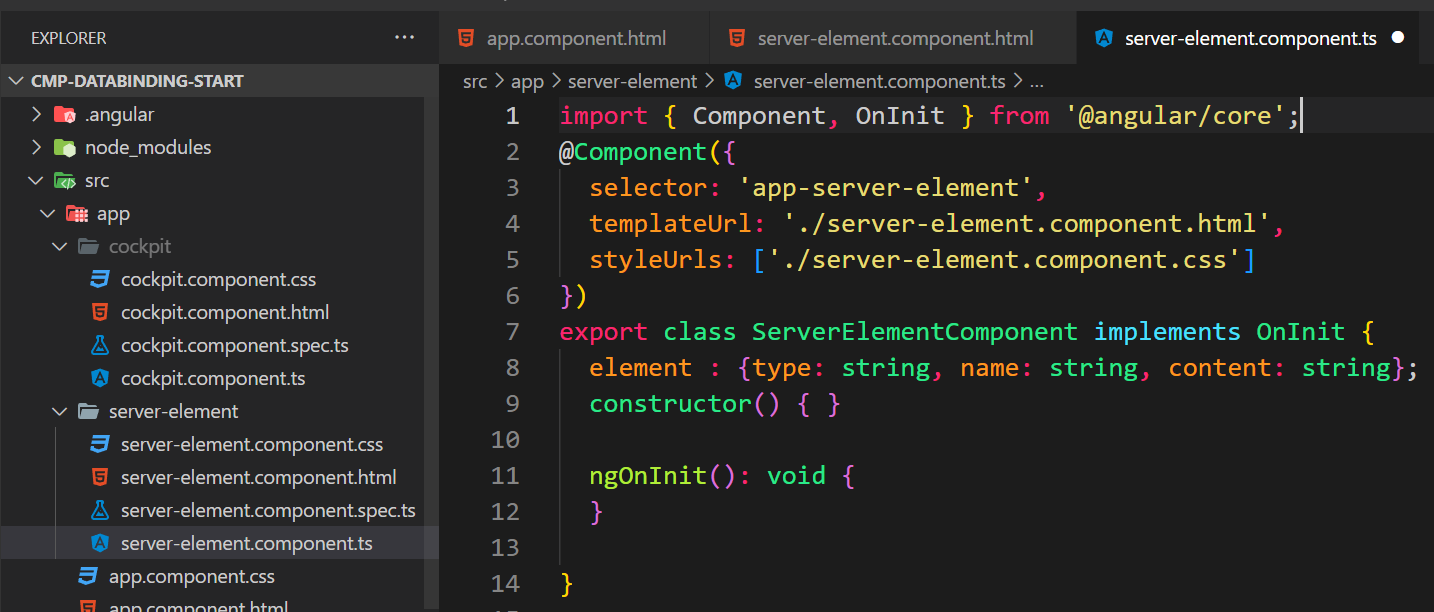


Where we can use property binding:



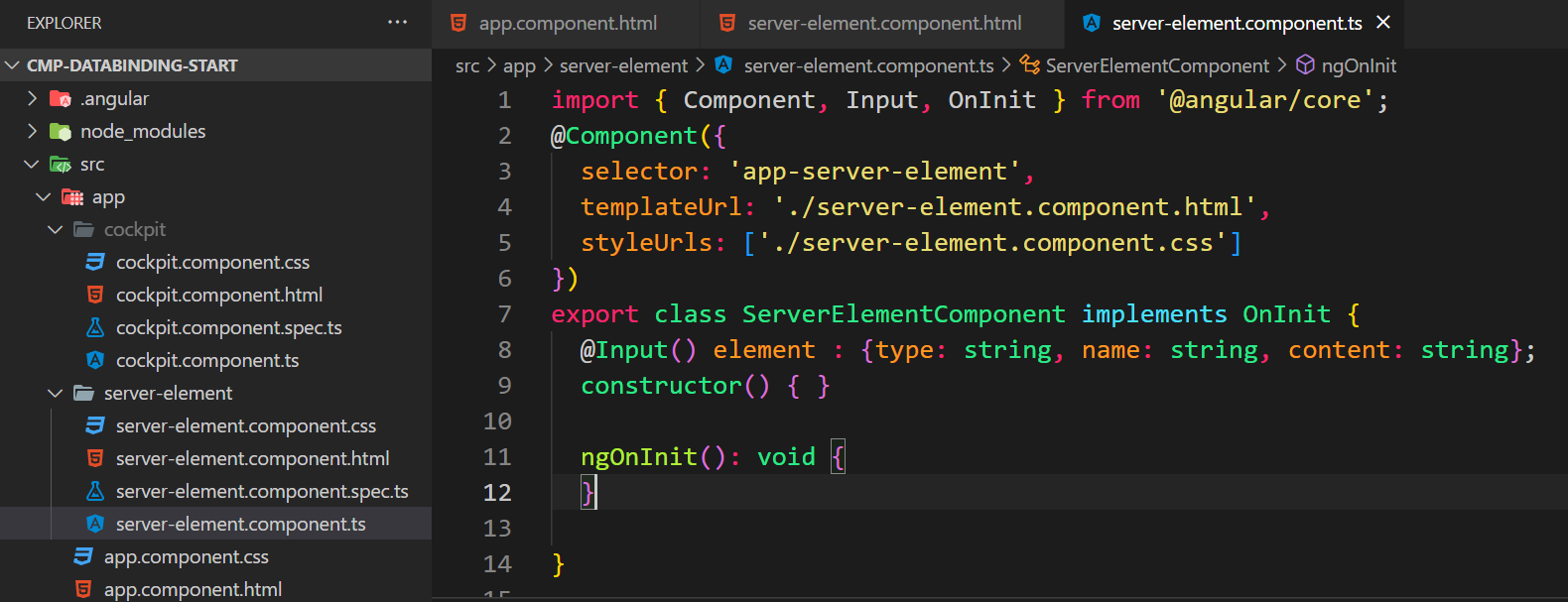
Next thing we will learn is how to share properties between the component.

Our server-element has this property called element:



We want to use this property in app-component.

To do so we need to add a decorator on this property. Once the decorator is added, we can use this property in other components as well.

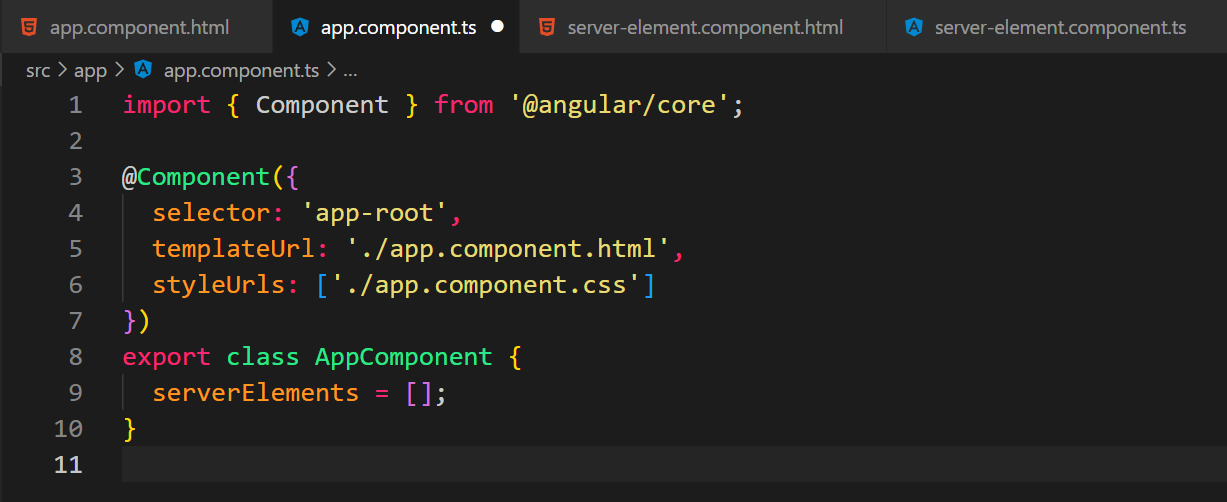


@Input() is imported from Input in angular/core.

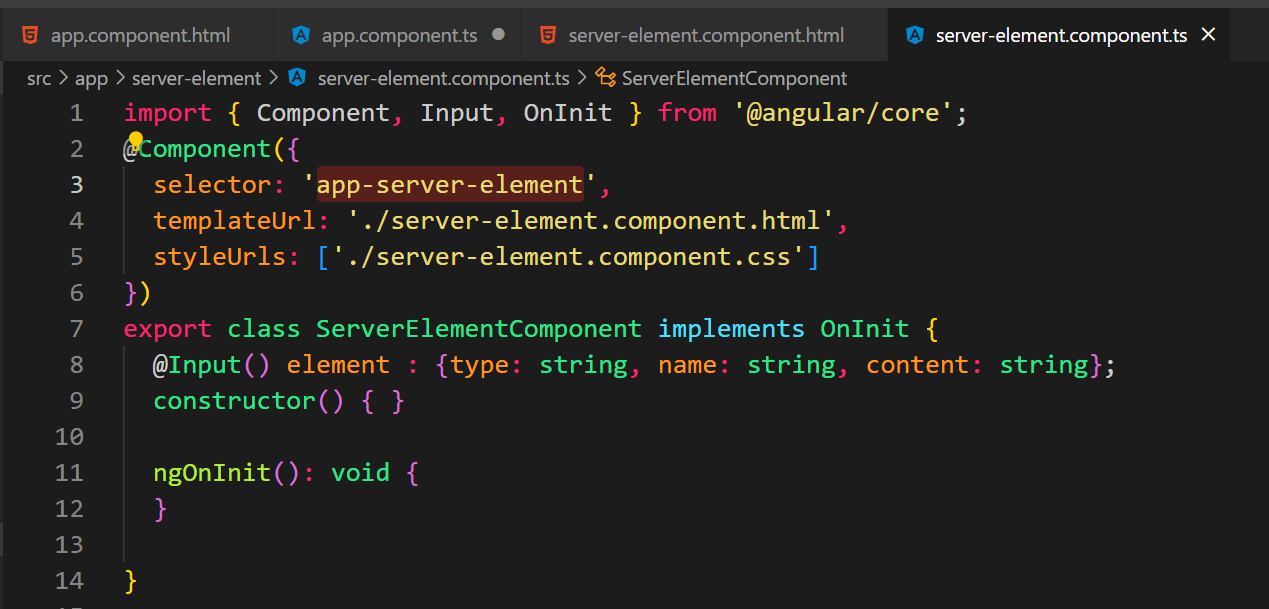
Now any element which is using this component through its selector, will be able to bind to this property called element.

See the app component TS:

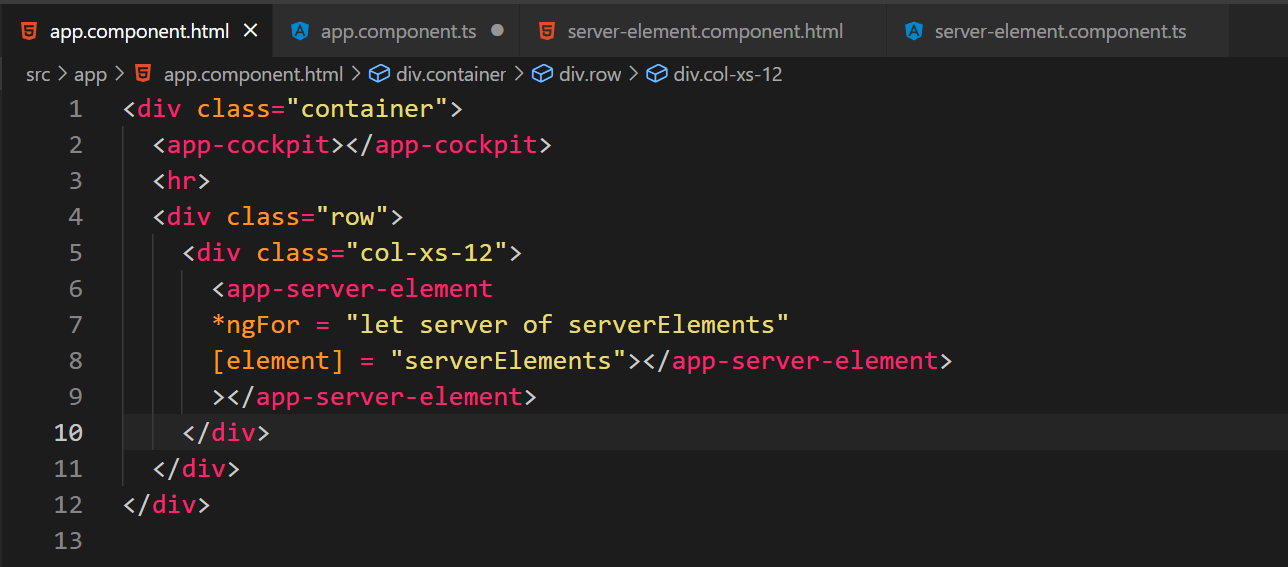
It has a property called serverElements:



Server-element component has a property element which is of type JS object having three things:

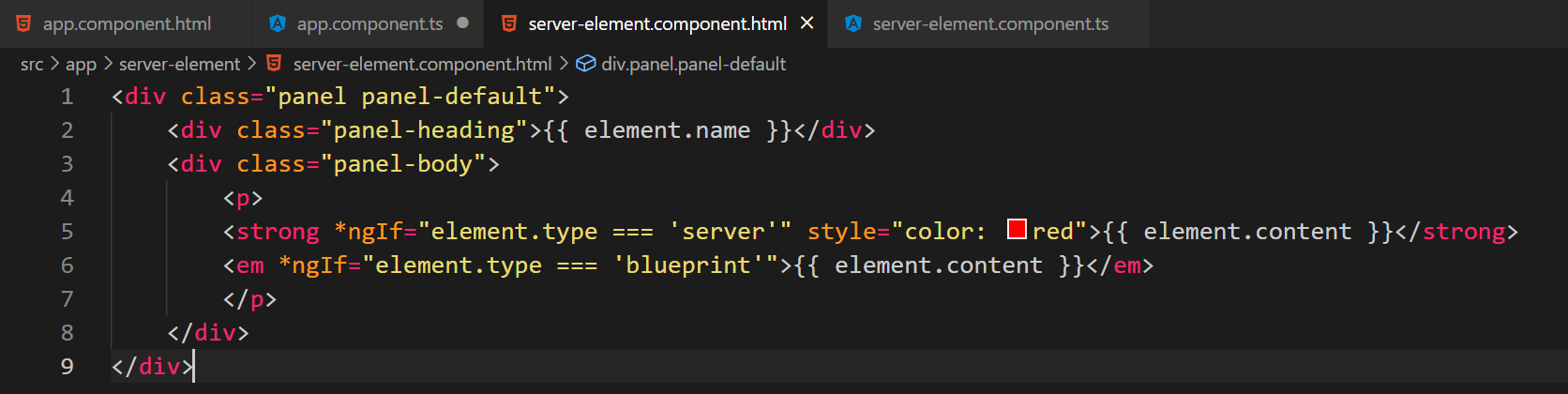


When we put the selector of this component inside the app component:

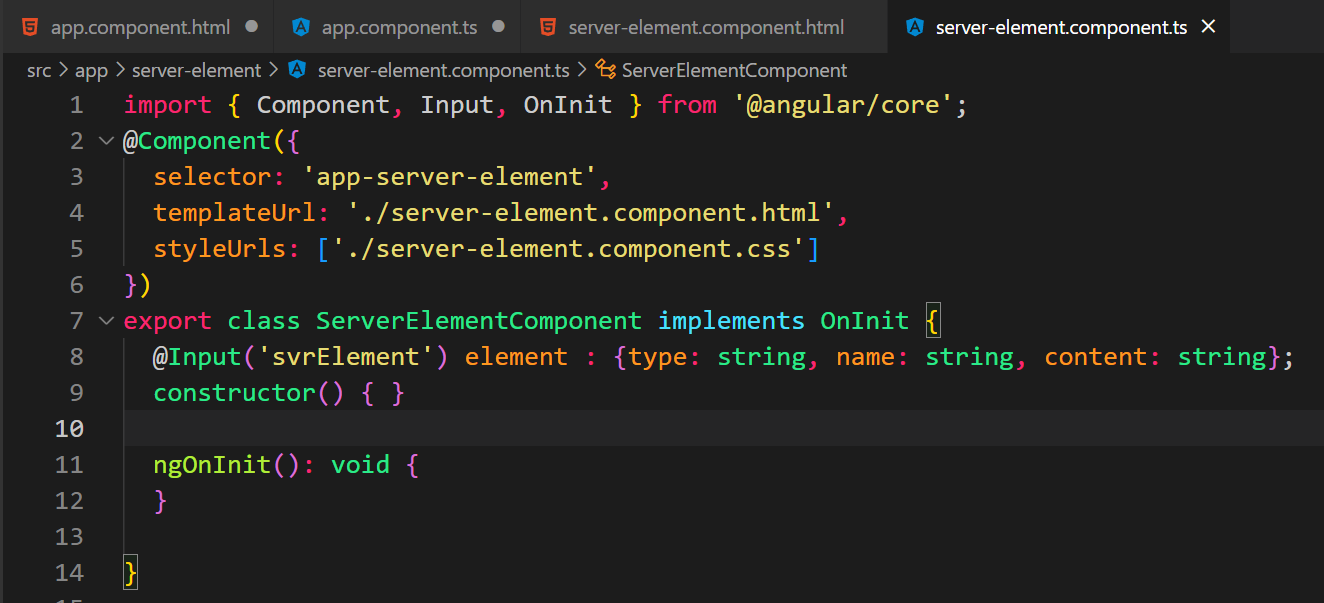


We here are binding element property of server component with the serverElements property of app component.

This is server components html:

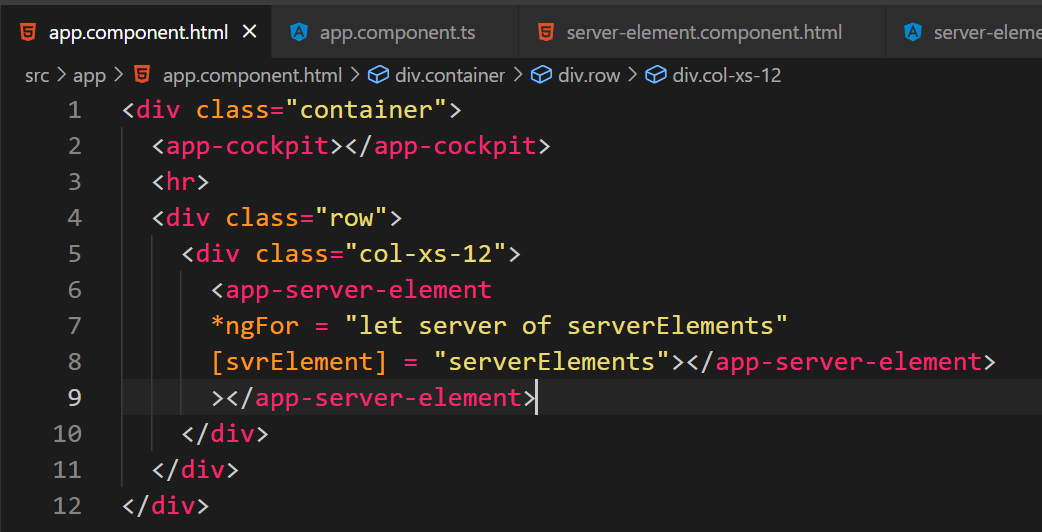


If you want to assign a custom name to this property in the parent component (different from what its name is in the original component):



Now we need to use svrElement to bind this property.

See the binding in app component html:



This method is used to pass data from a parent element to a child element.

Read more here: <https://angular.io/guide/inputs-outputs>