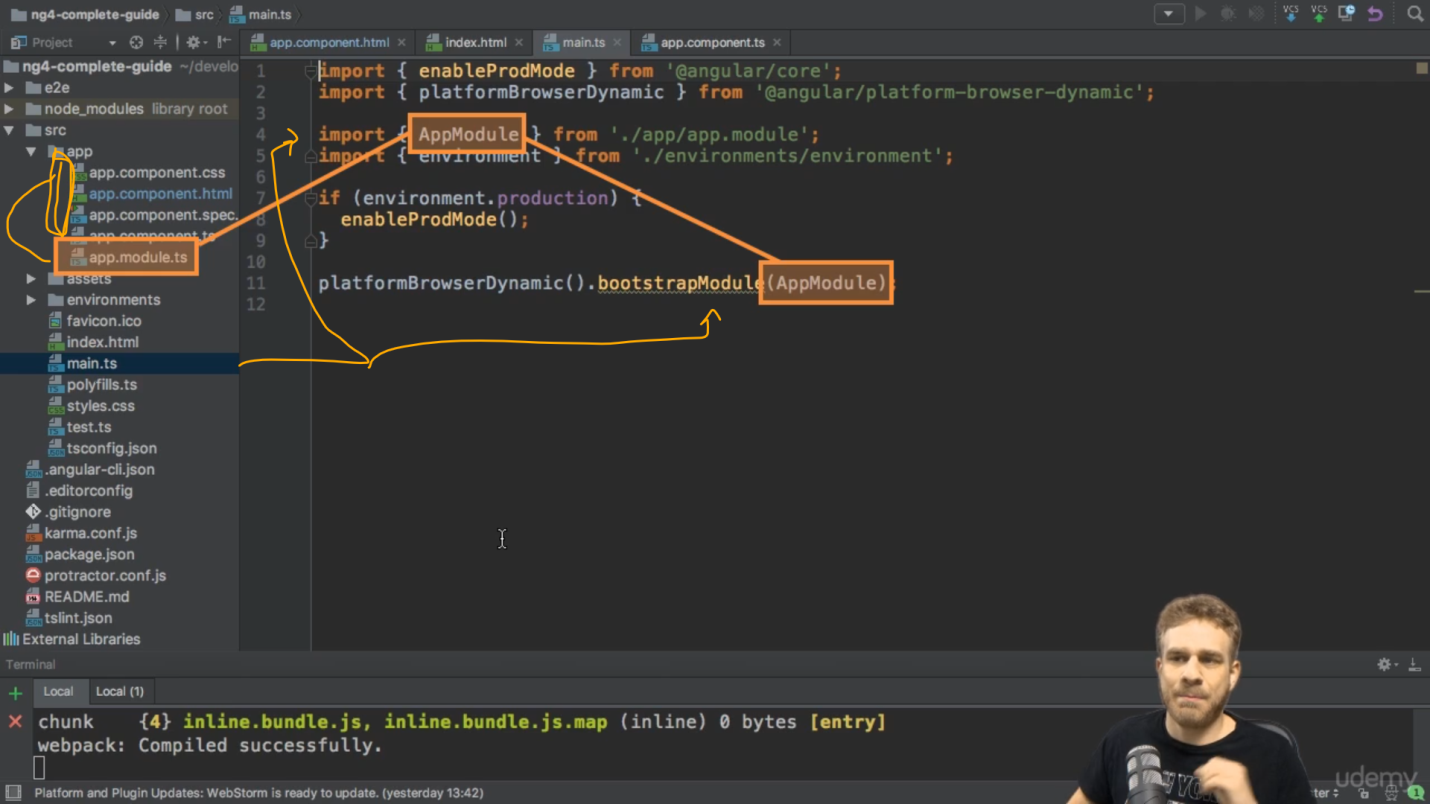
Working :

First of all Main.ts is executed, and in that they bootstrap app.module where components are declared.

These components have selectors defined which can be used in index.html ( the face of our website, the only page that is being showed on website) as tags.

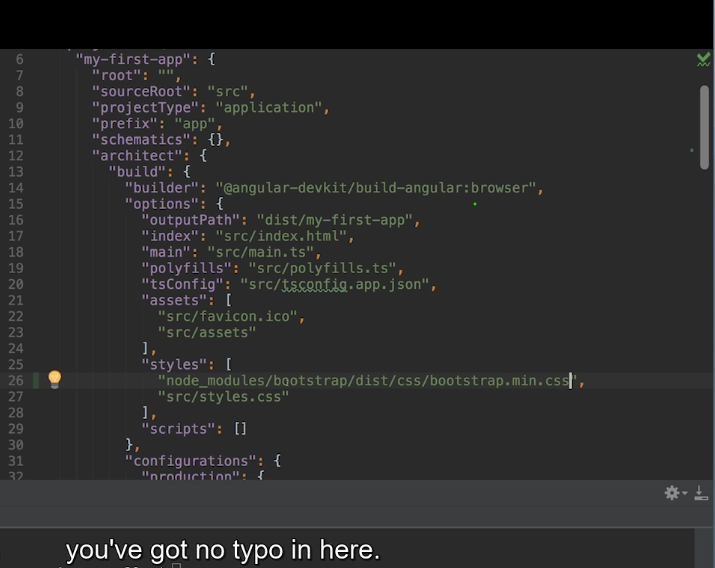


If we add more components in the project, we can use their selector tags in the app component.html and not in index.html …. Coz onlt

ng new project name to make new proj

To include bootstrap in the project locally use npm install --save bootstrap@3

And in angular.json file add the location of bootstrap file in styles section



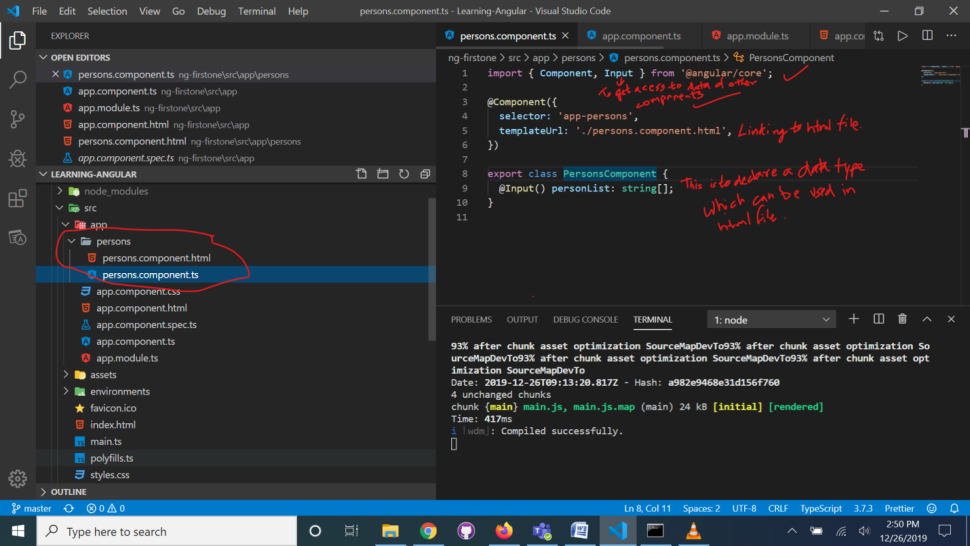
To add a new component, make a new component and html file, preferably inside a new folder.

Every component has selector field which assigns it a name which can be used as a tag in the html file.

The html file of that component is then connected with the app component by mentioning the new component’s tag in its(app component) html file which in turn has a selector in its own component.ts, which is called in index.html !

1.New folder -> Make new.component.ts and new.component.html

2. New.Component.ts -> add @component and add selector name -> add export class with any name which will be imported by app.module.ts

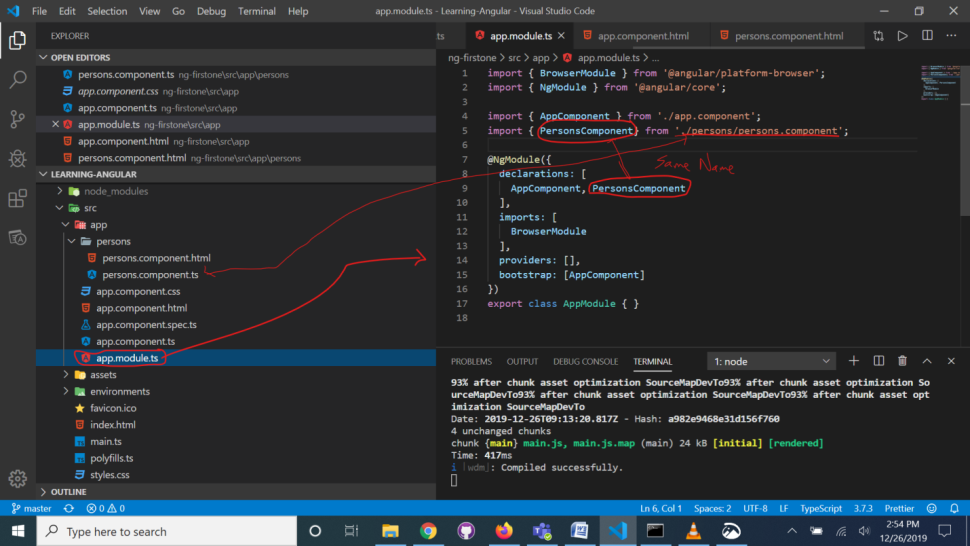


*Remember the name used in export statement is used in all other files for referencing this file.*

*Here we chose PersonComponent.*

*Also, we declared the personList as the input, so this variable will be given some input of string array type in the html page.*

3. App.module.ts-> import the component and add its name in declaration.



OR:

* Use ng g c person -it -is command to create components, this automatically creates a folder and adds all the files to it and add connections as well to the app.module and other files.

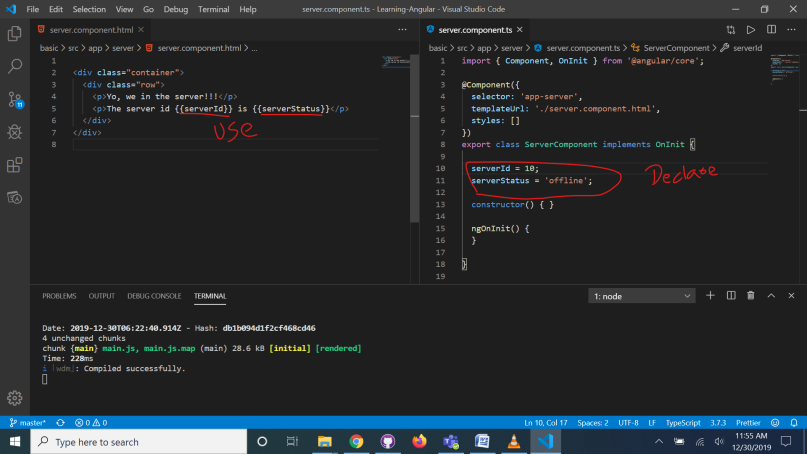
In app.component.ts we can have selector as a tag ( <> ), an attribute ( [] ) or even a class (.)

**Data Binding :** communication between typescript code and html…….

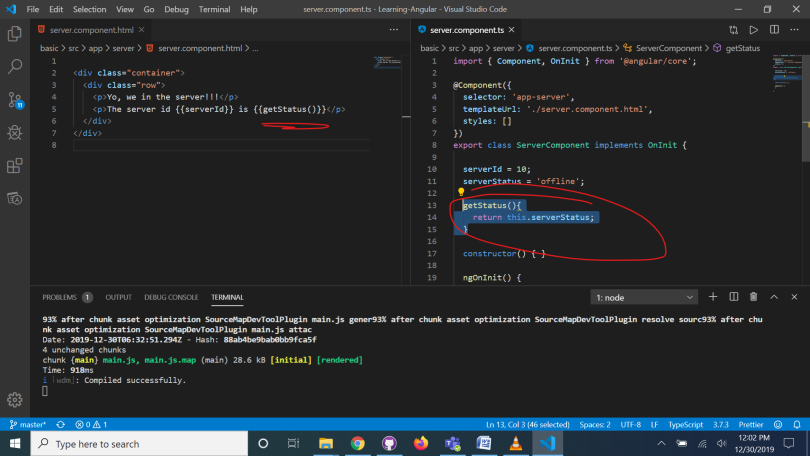
1. **Output Data :**

String Interpolation {{data}} – this basically return string… (converts int into string and returns)

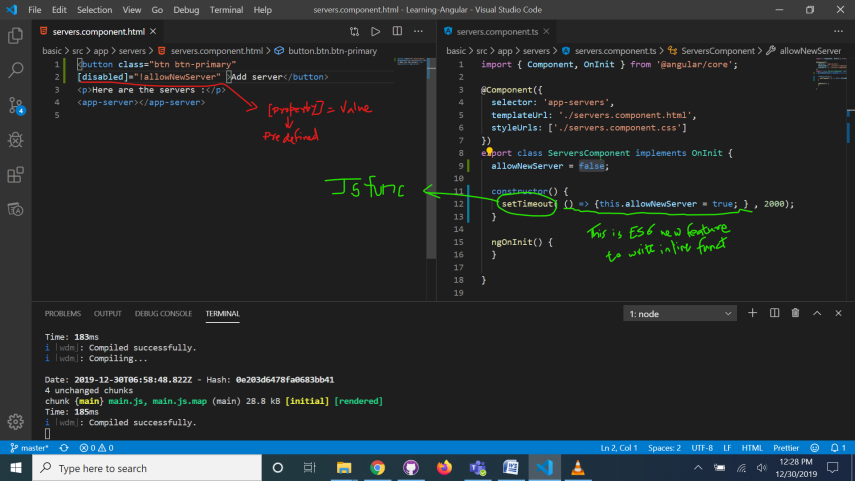
It can have variable and raw string and method calls which returns string .



It can have variable and raw string and method calls which returns string for eg :

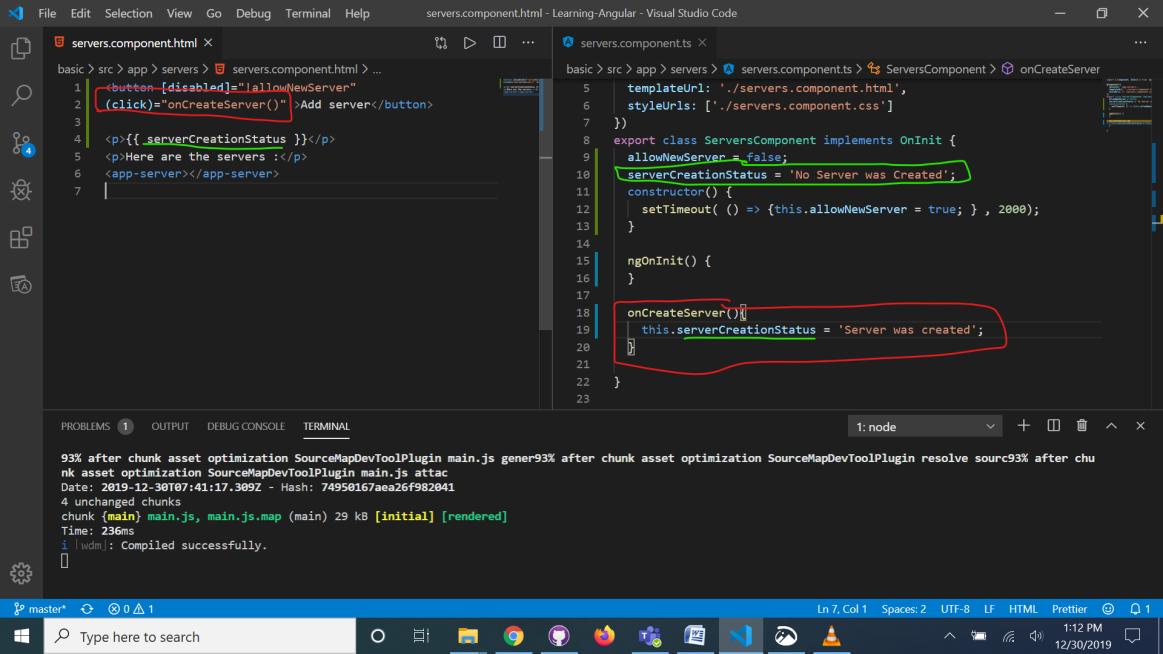


property Binding [property] = “data” –



1. **React to user event:**

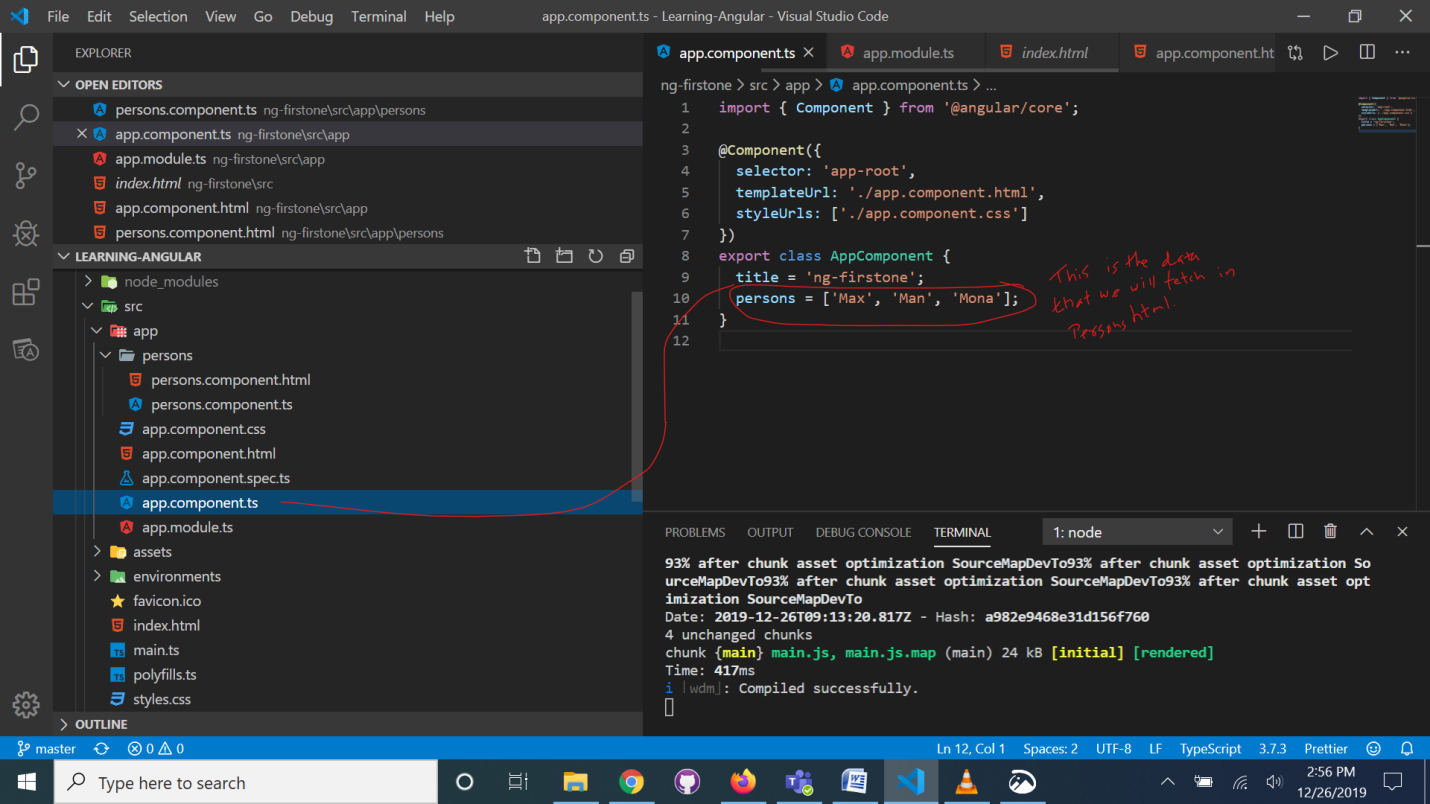
Event Binding (event) = “expression”



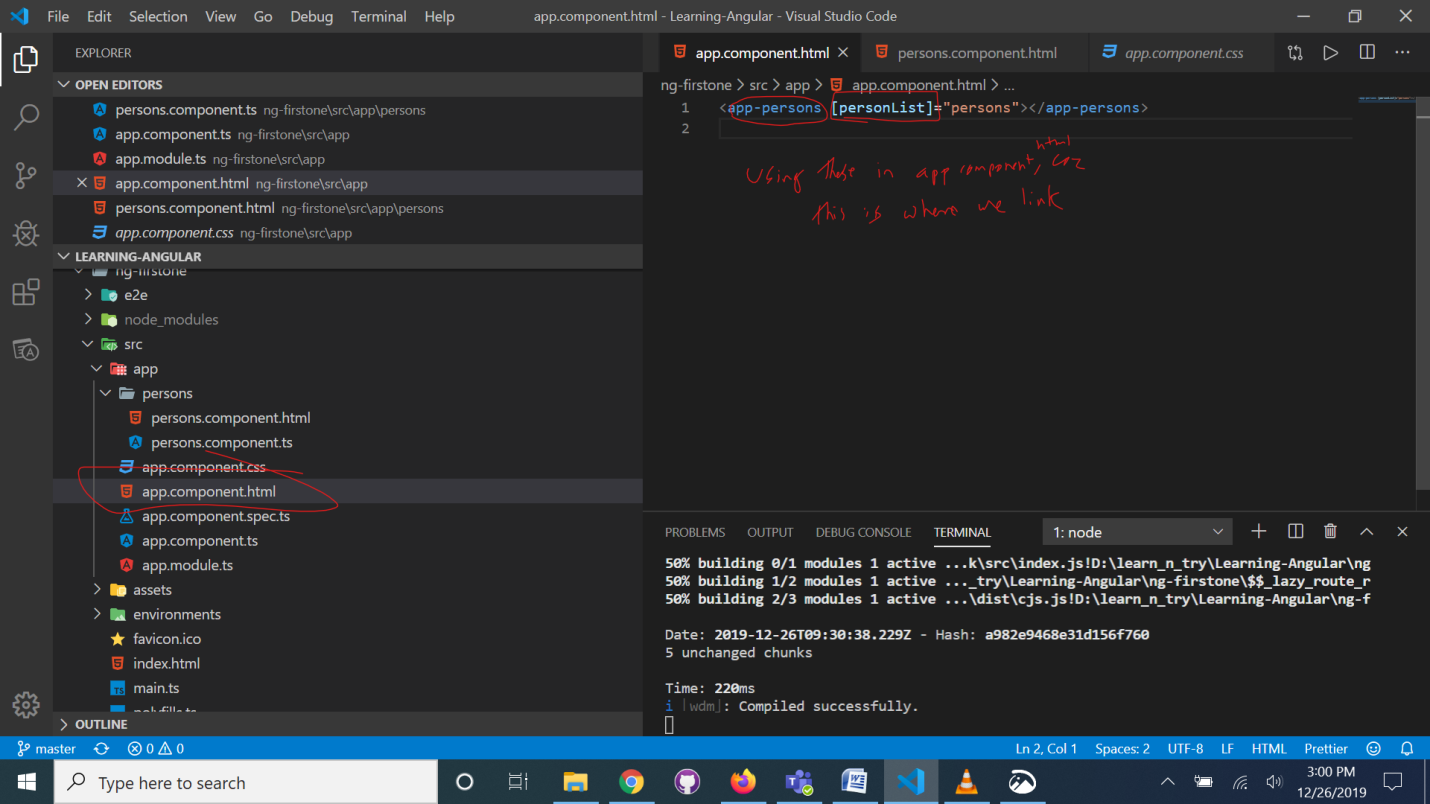
1. **Both way:**

Two – way data binding [ (ng-model) ] = “data”

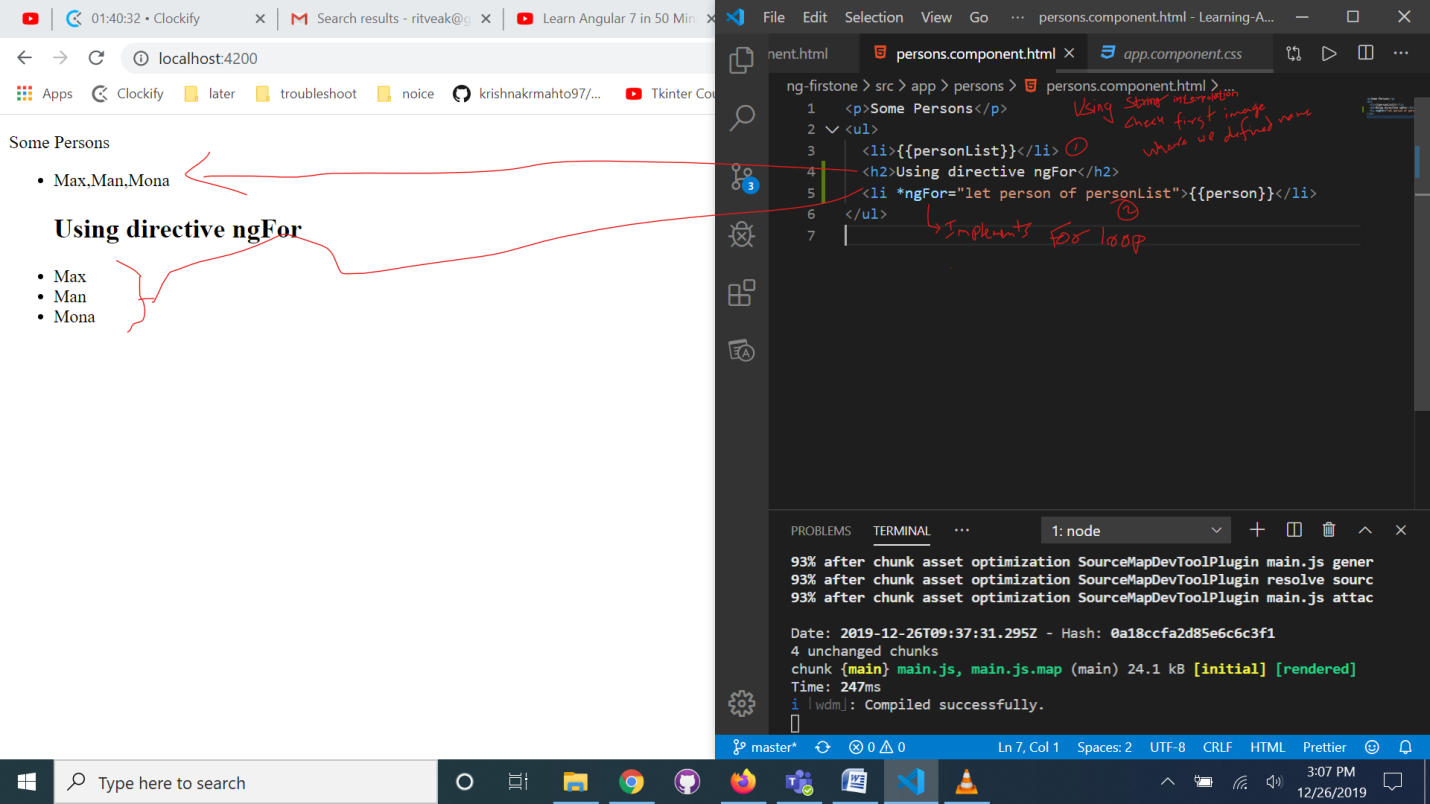
4. Either declare a data in app.component’s export statement or get data from http req, either way it is kept in export section.



5. Use the new component’s tag in app.component.html *(The html file of that component is then connected with the app component by mentioning the new component’s tag in its(app component) html file which in turn has a selector in its own component.ts, which is called in index.html !)*

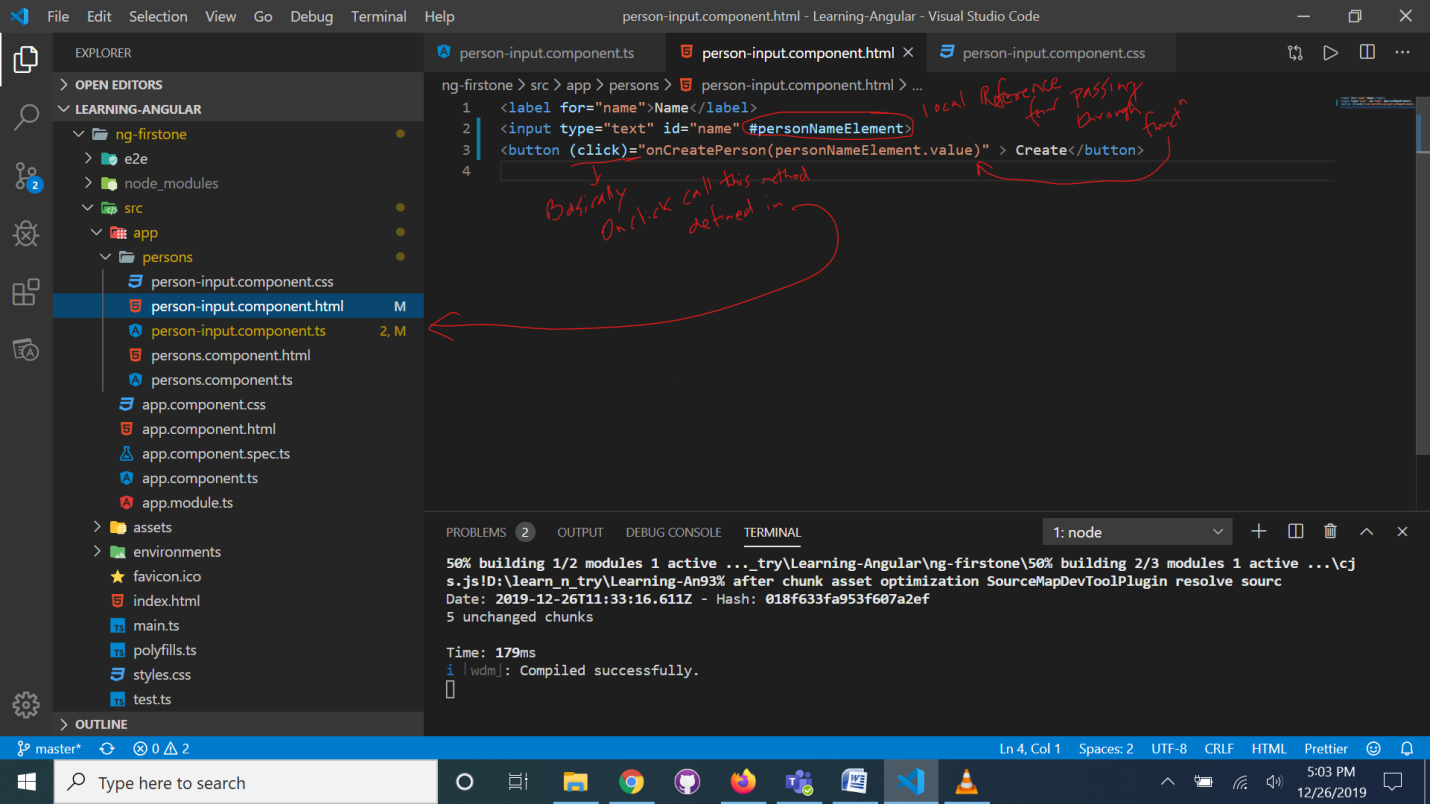


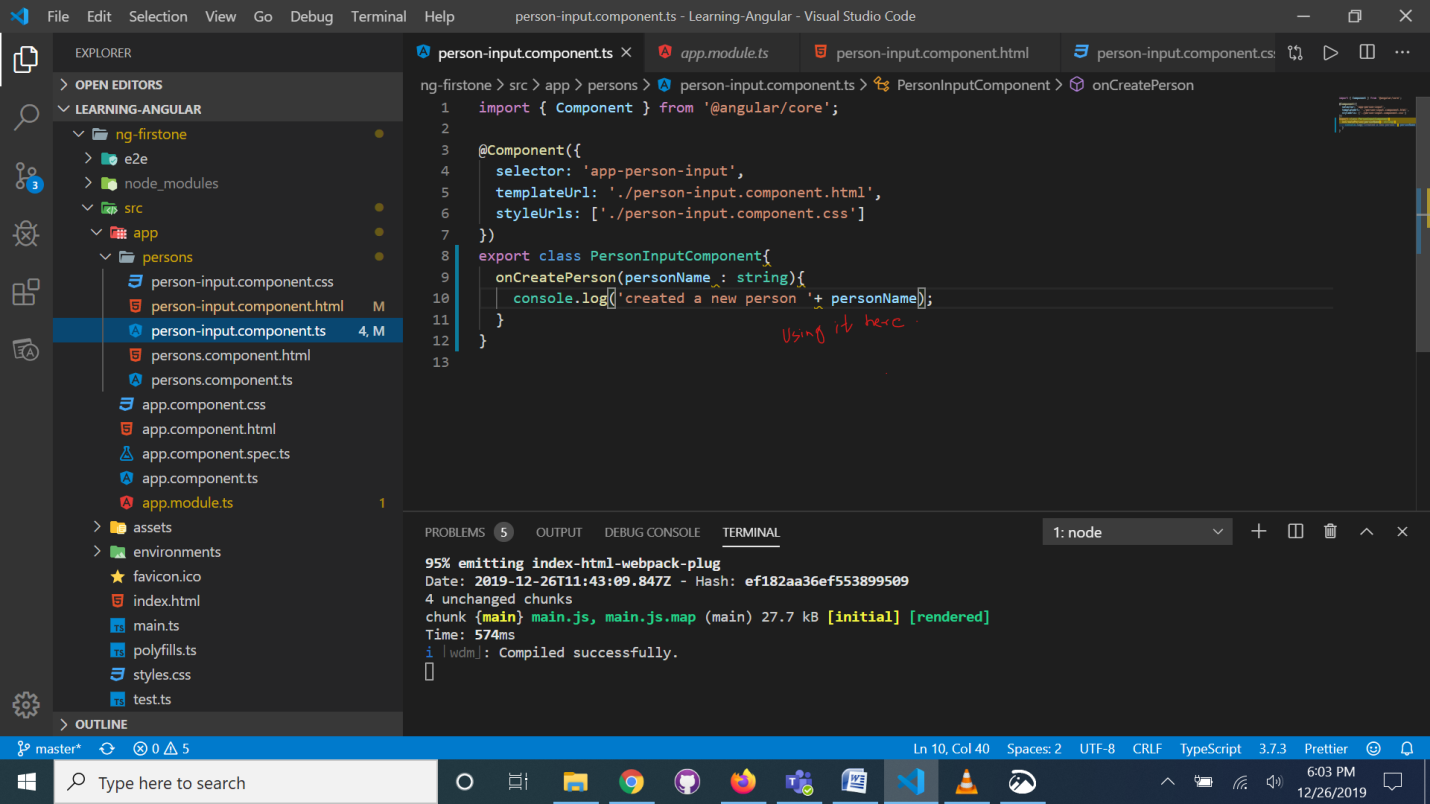
6. Using ngFor we can make a for loop fro displaying elements of array.



TAKING INPUT AND ADDING IT TO THE LIST THAT WE HAVE:

Make three new files (.ts, .html, .css)and connect them as we did earlier(by importing and declaring in app.module.ts):





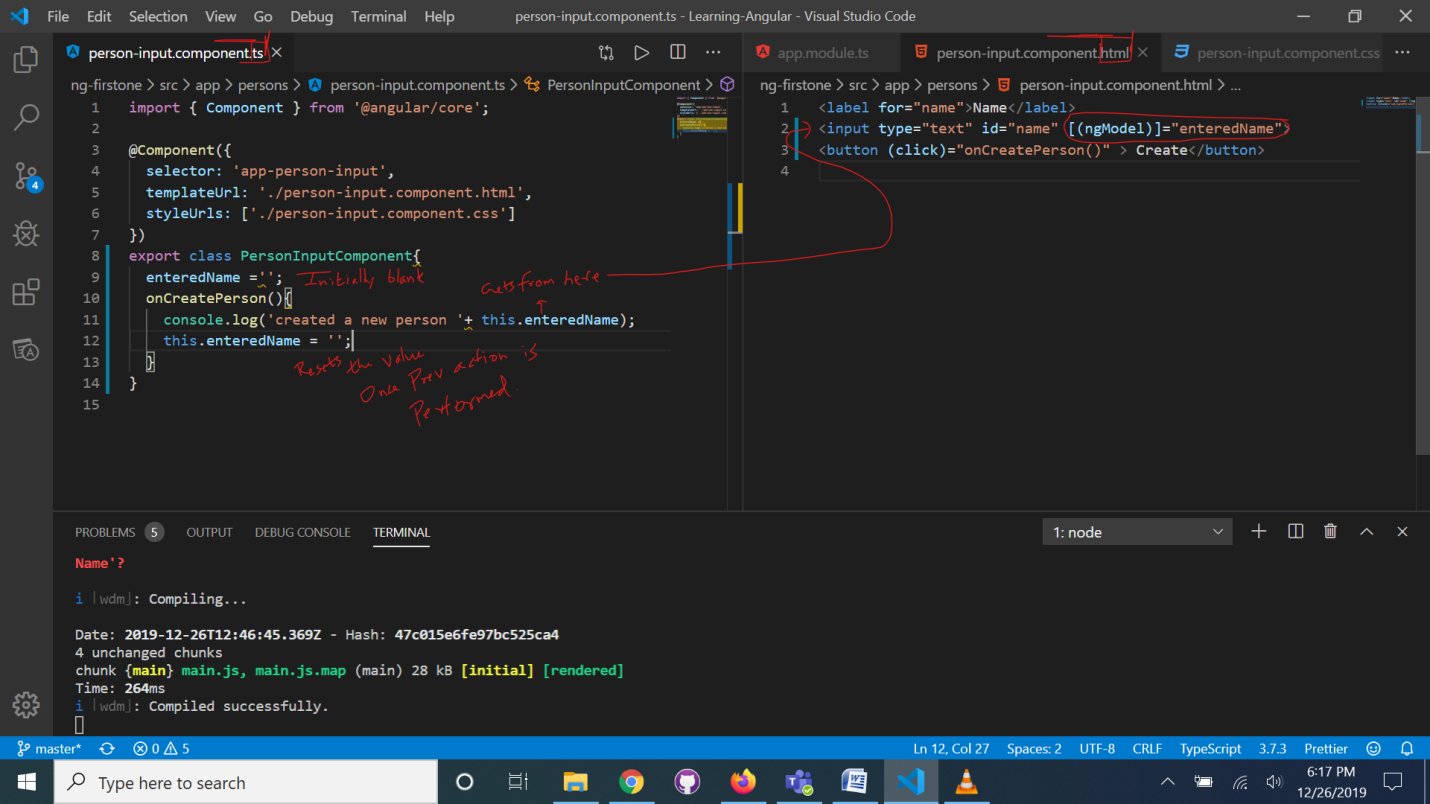
Property binding -> [] eg [personList]

Event binding -> () eg(click)

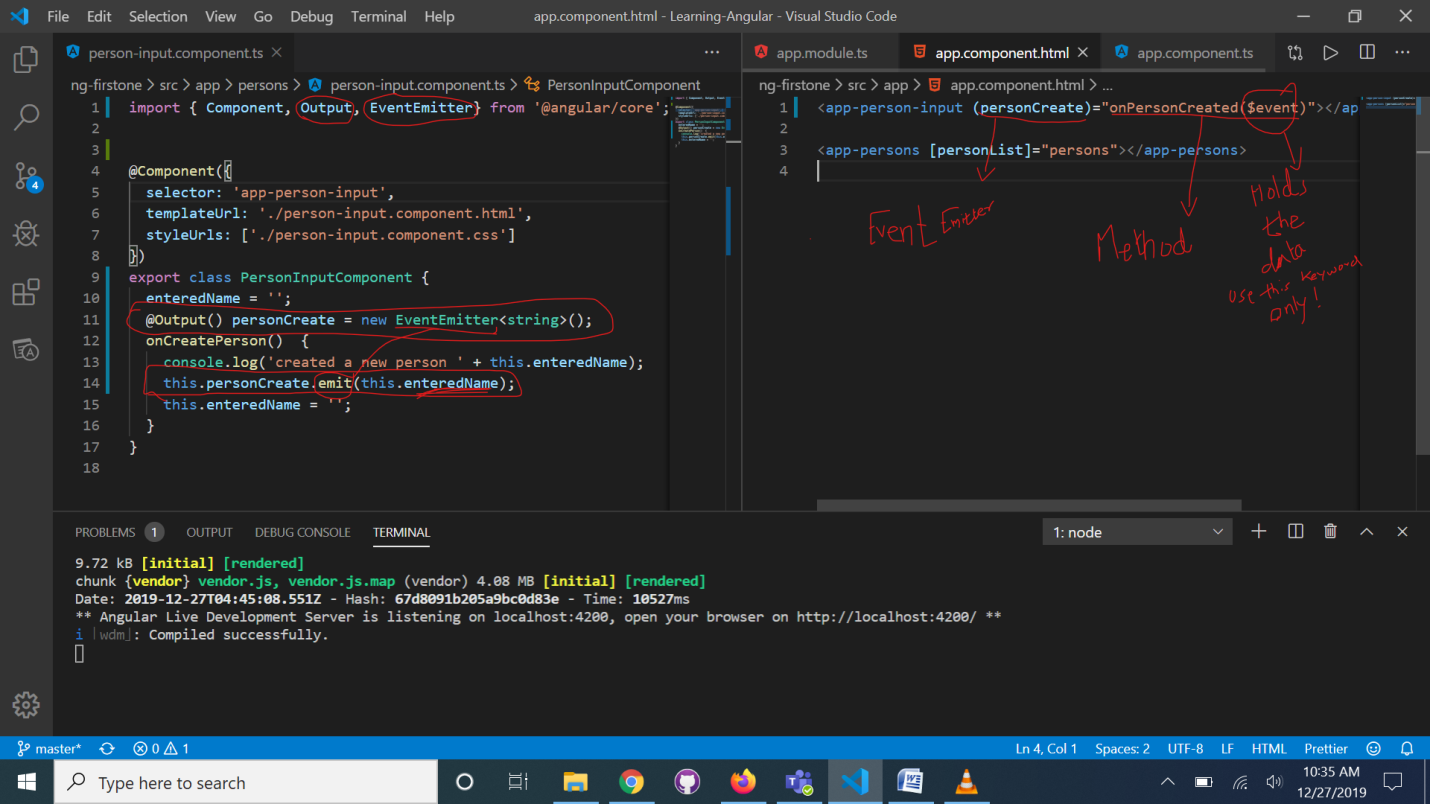
String interpolation -> {} eg {{person}}

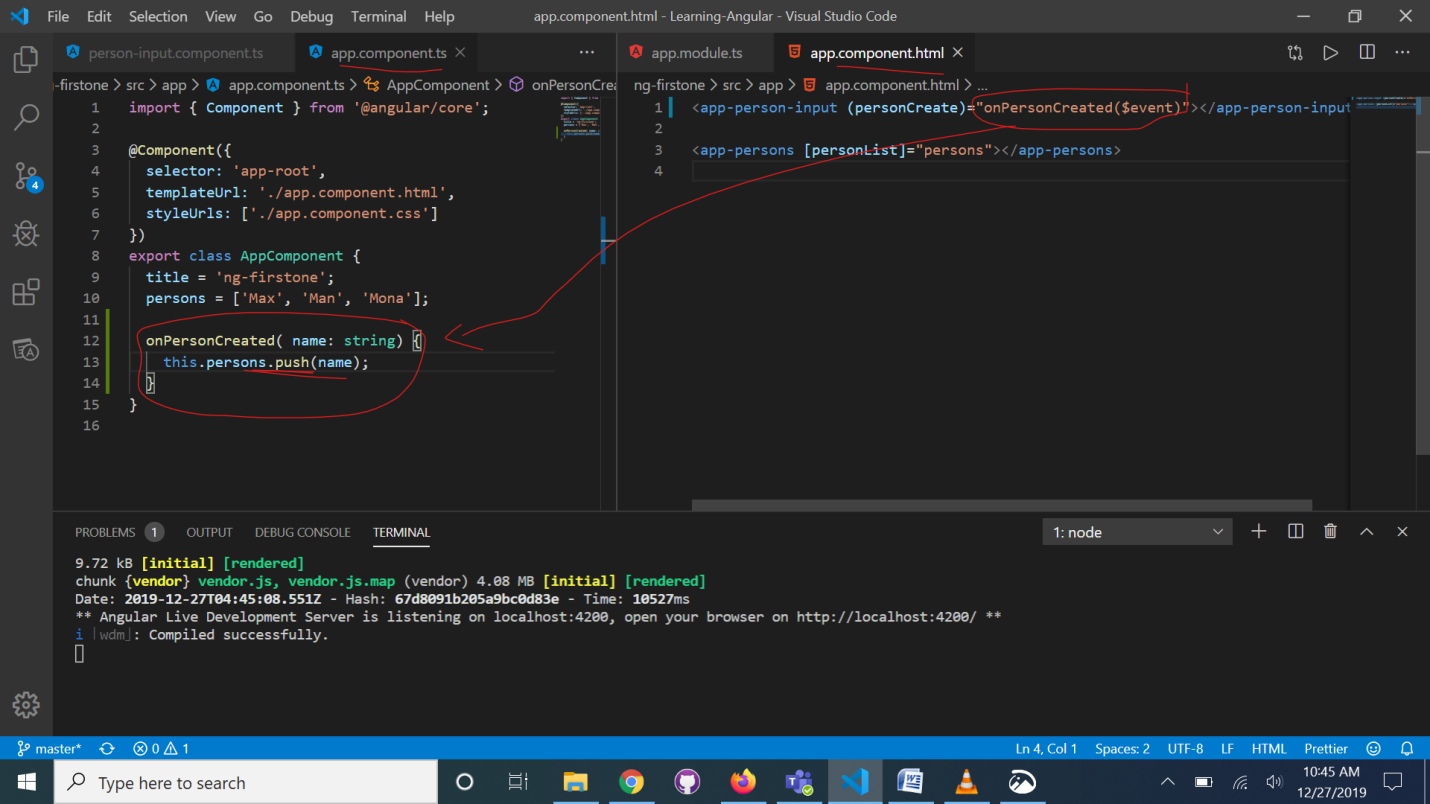
Two way Binding->[()] eg [(ngModel)]

Basically we can change the value of the variable by entering a value and we can reset the value by our code.



For our own property binding we used Input as a decorator.  
For our own event binding we use output as a decorator.



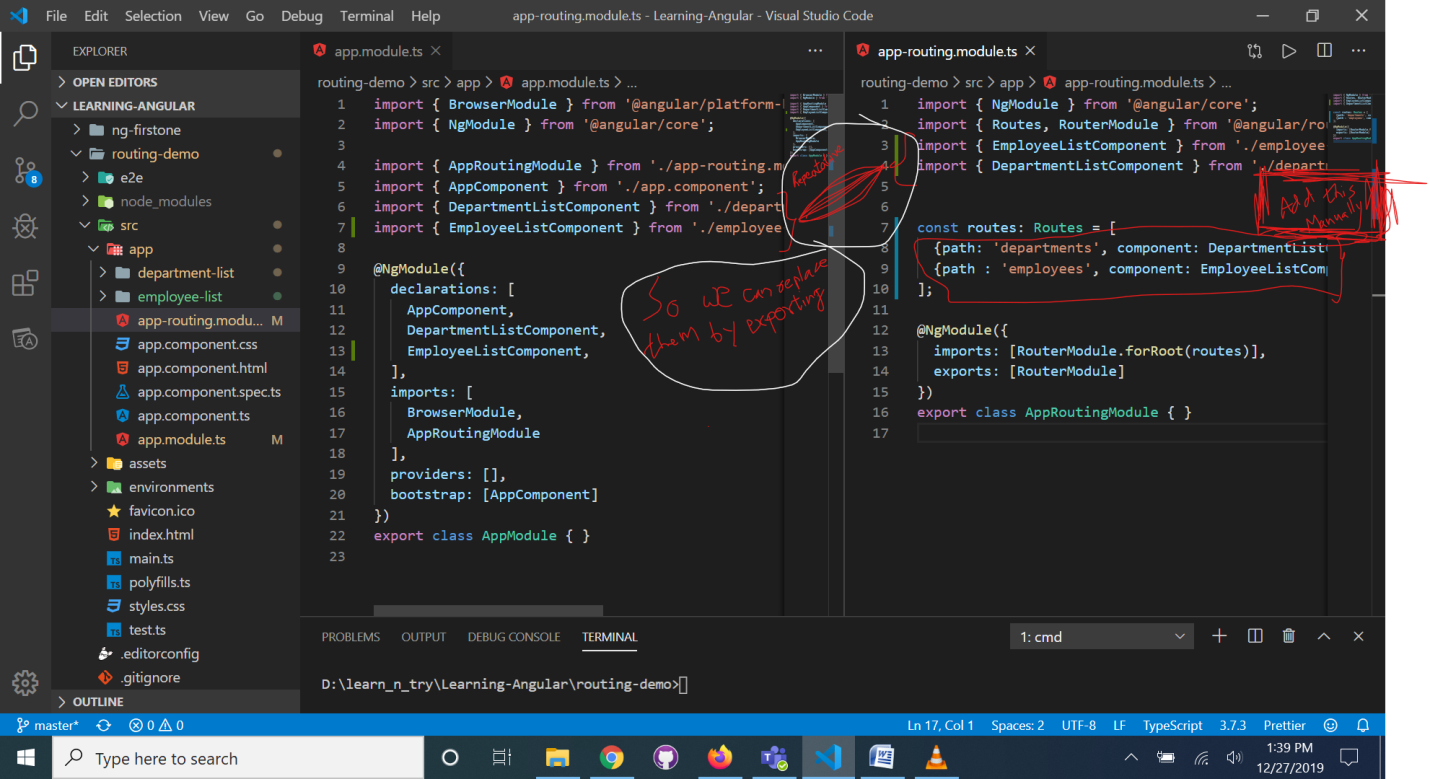


ROUTING:

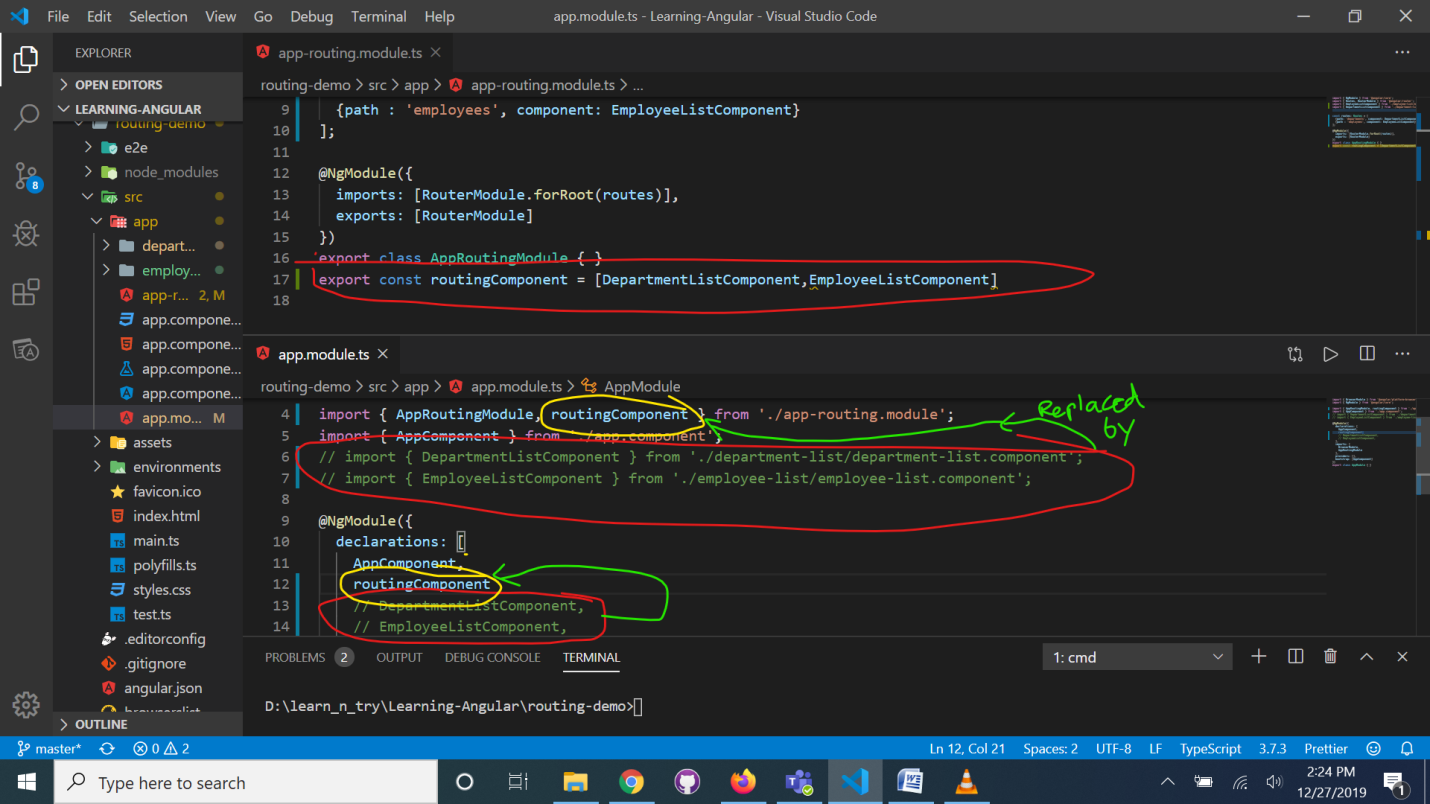
The example below doesn’t use services, they have hard coded stuff hence doesn’t require…. Go below to original example of our’s to understand services

Separate files, new angular proj . <https://www.youtube.com/watch?v=Nehk4tBxD4o> refer this when stuck.

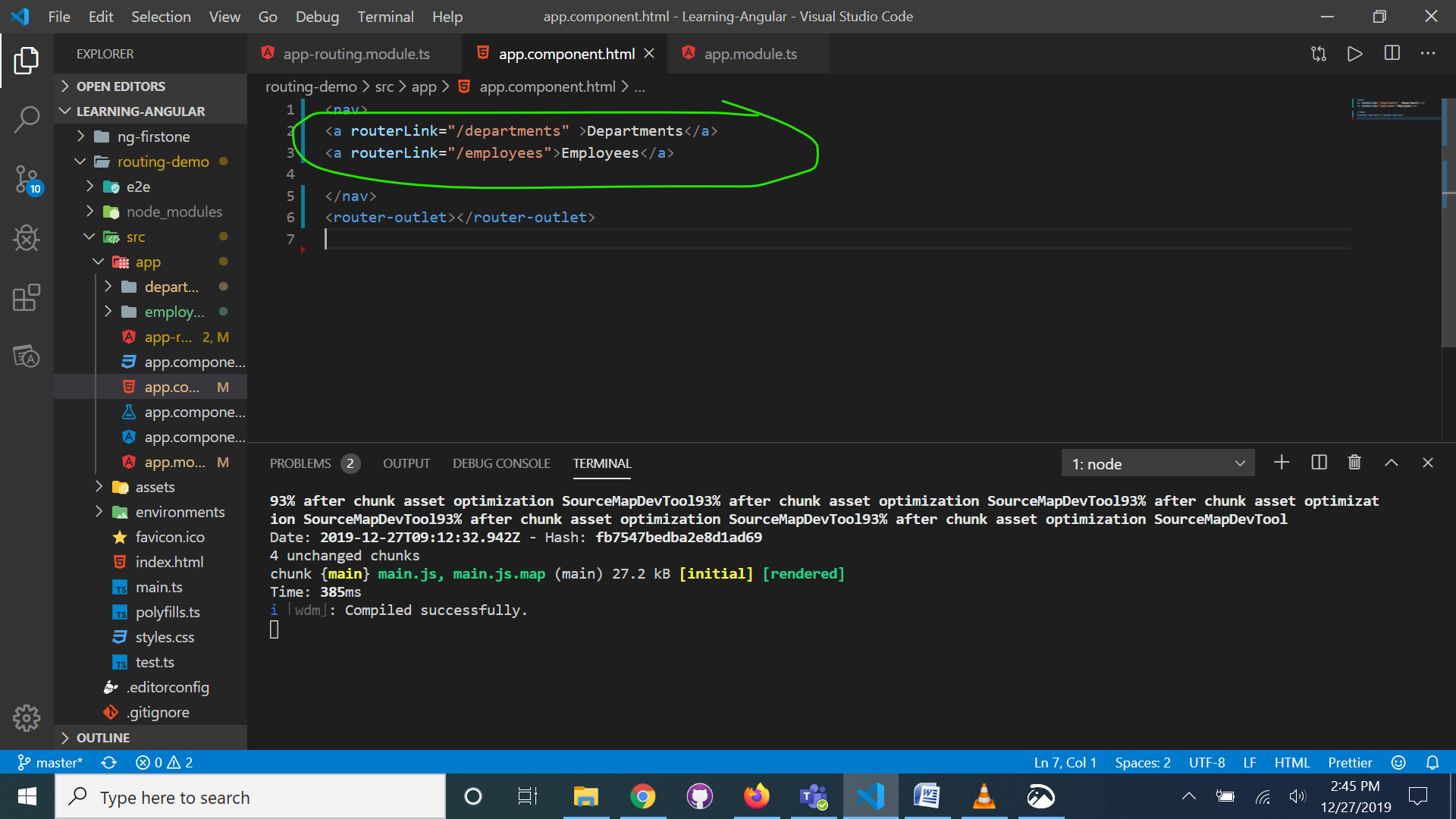
* We can use “ng g c department-list -it -is” command to create components, this automatically creates a folder and adds all the files to it and add connections as well to the app.module and other files.



Now no need to declare any components manually in module.ts, just add them to routing module:



Without doing this we have to manually go to the url to see each route.



Now coming back to our example, We can use services to support router