1.a Course Name: Angular JS Module
Name: Angular Application Setup

Step 1 : Before install angular we need to install node js and vs studion
Step 2 : Open command prompt

Use this commands for install angular

- **npm install -g @angular/cli**

  above command is used to install the angular

- **ng v**

  through above command we can check the version of angular

Angular CLI: 16.1.4
Node: 18.15.0
Package Manager: npm 9.5.0
OS: win32 ia32

Creating a Angular Application

F:\Aditya_College_Informations\meanstacklab\Module-2\Angular>ng new myapp

Now let see how build the server

Step1 : place mcart folder in any Drive (C,D,E,F….)

Then use below commands

D:mcart > npm install

This will create a folder called node_modules with all the dependencies installed inside it

After complete the installation check all node modules are installed or not

Then run the mcart using below command

D:mcart>ng serve --open

1B . Module Name : Component

Create new component called hello and render hello angular on the page

Angular app – One more modules

Module – One or more components and services

Components – Html + css

Services – Business logic

Module interact and ultimately render the view in the browser

Let’s start the angular application is hello-world

Create angular folder

E:\Angular>ng new hello-world

//Above command for create angular application

E:\Angular>ng serve –open

//Execute angular application

Then open it visual studio go to src->app->app.component.ts

Find title property add another called name

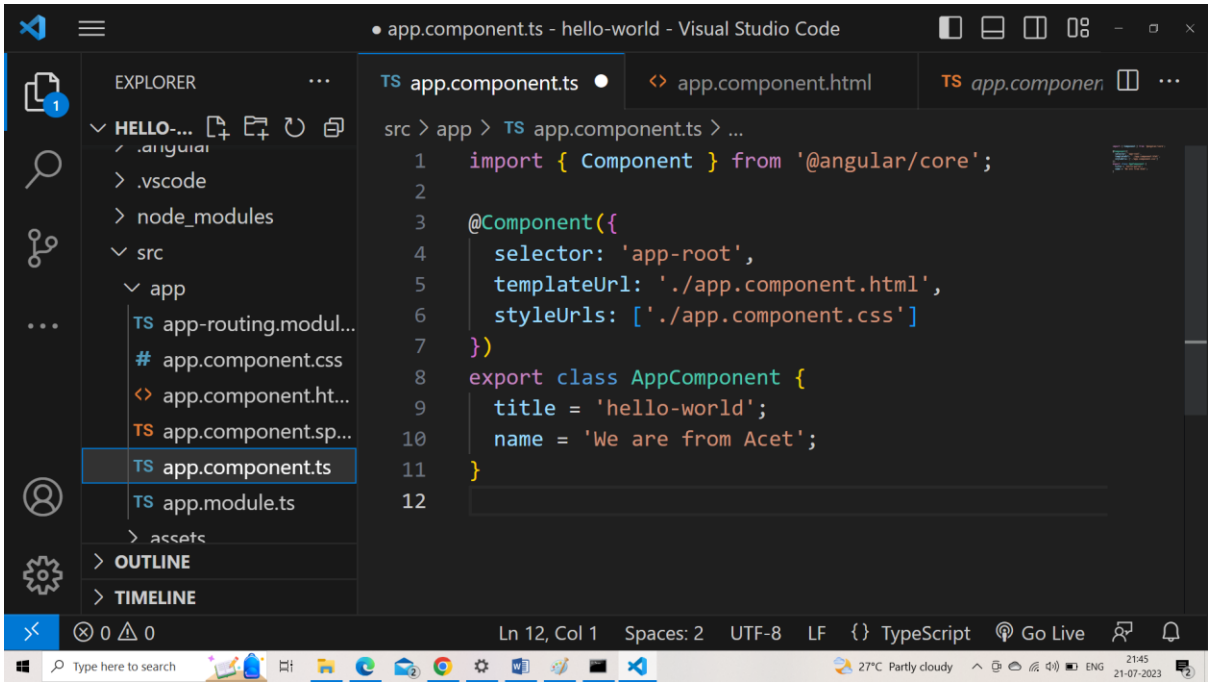
Then go to scr->app->app.component.html

Find the `<span>{{ title }} app is running!</span>`

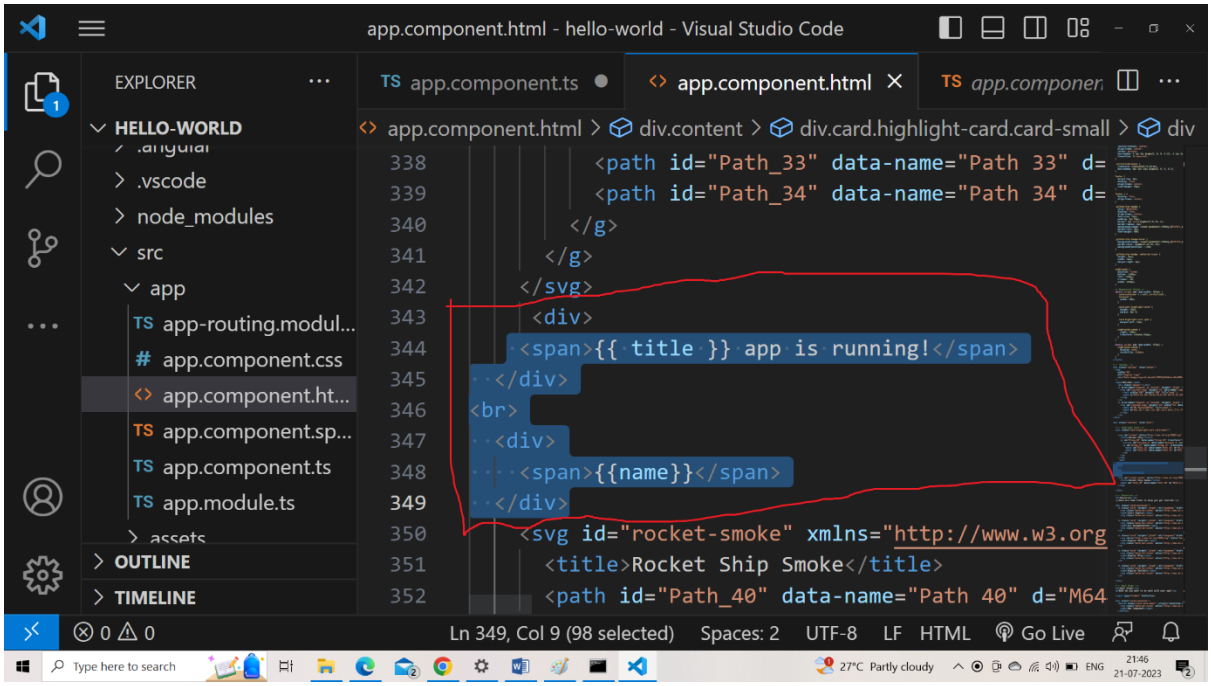
Then add another tag with name property

```
<div>
   <span>{{name}}</span>
 </div>
```

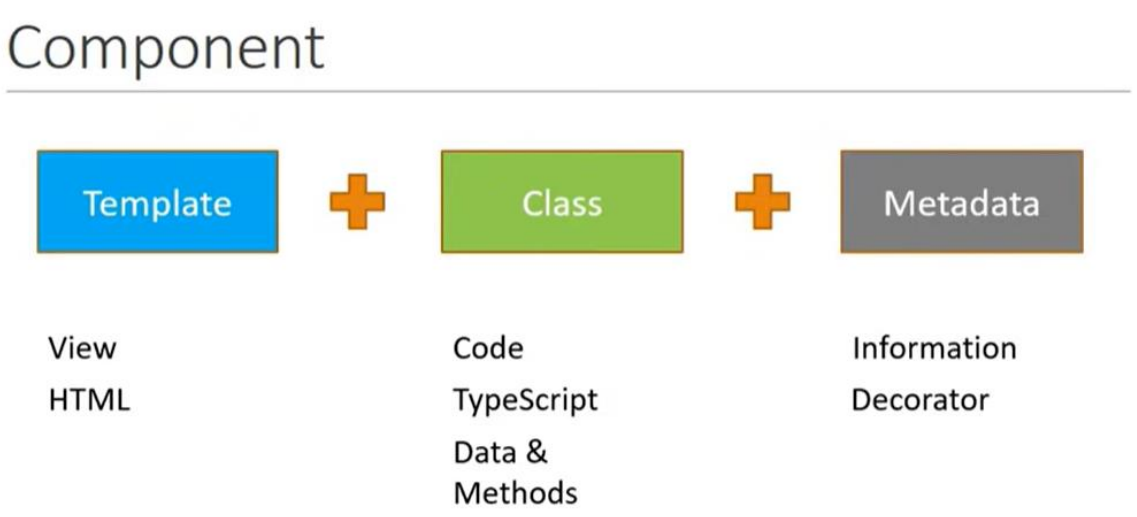
Component.ts



Component.html



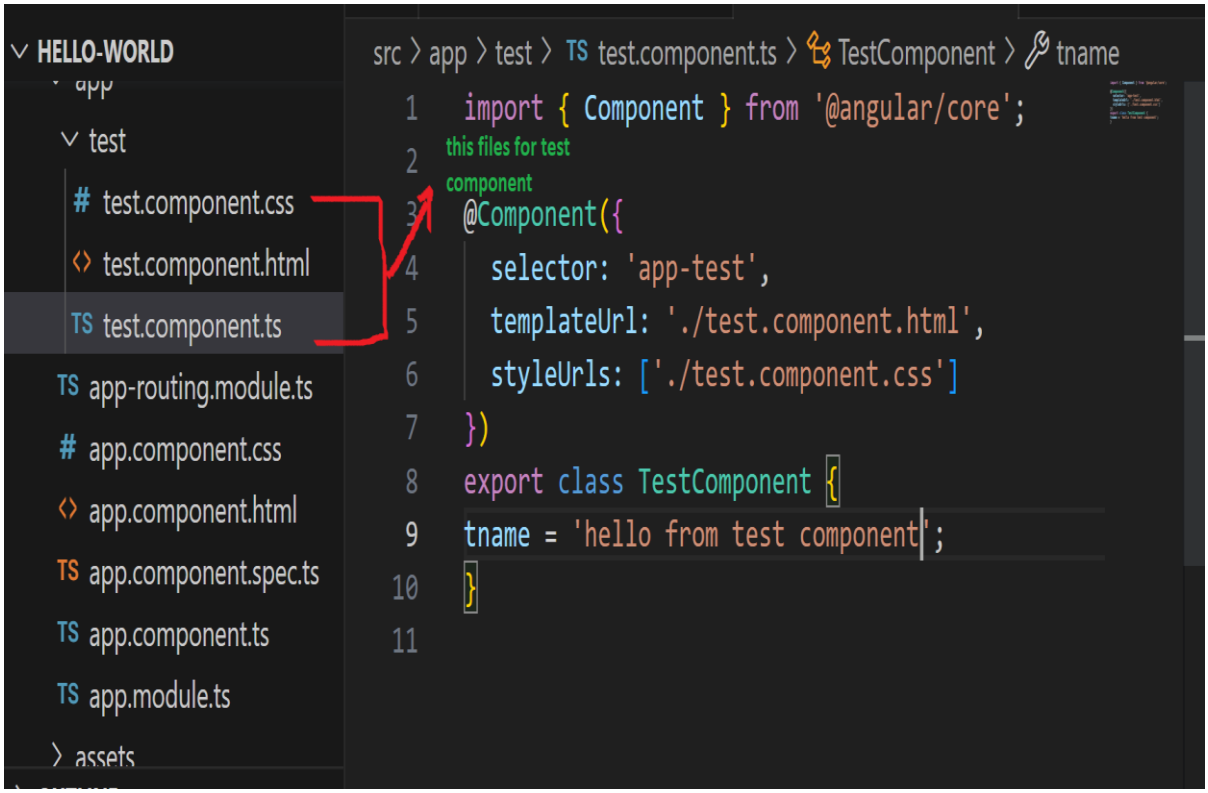
What is Component?



For creating user define component we need to use this command

>ng g c test

After execute above we find below files for test component

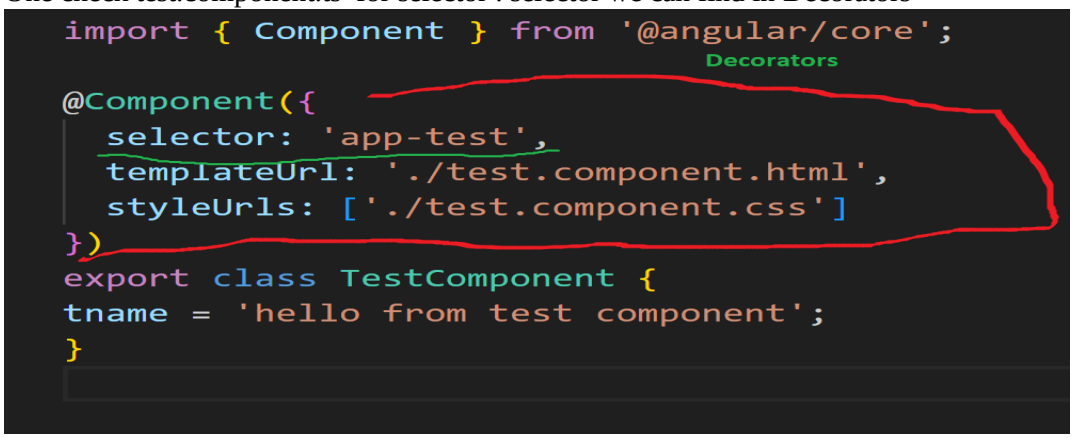


How to render hello from test component

1. declare tname variable in TestComponent class
2. now open test.component.html add below script

```
<p>test Component Works!</p>
<p>{{ tname  }}</p>
```

3. One check test.component.ts for selector . selector we can find in Decorators



4. Now we have to add that selector in app.component.html

** **

{{ title }} app is running!

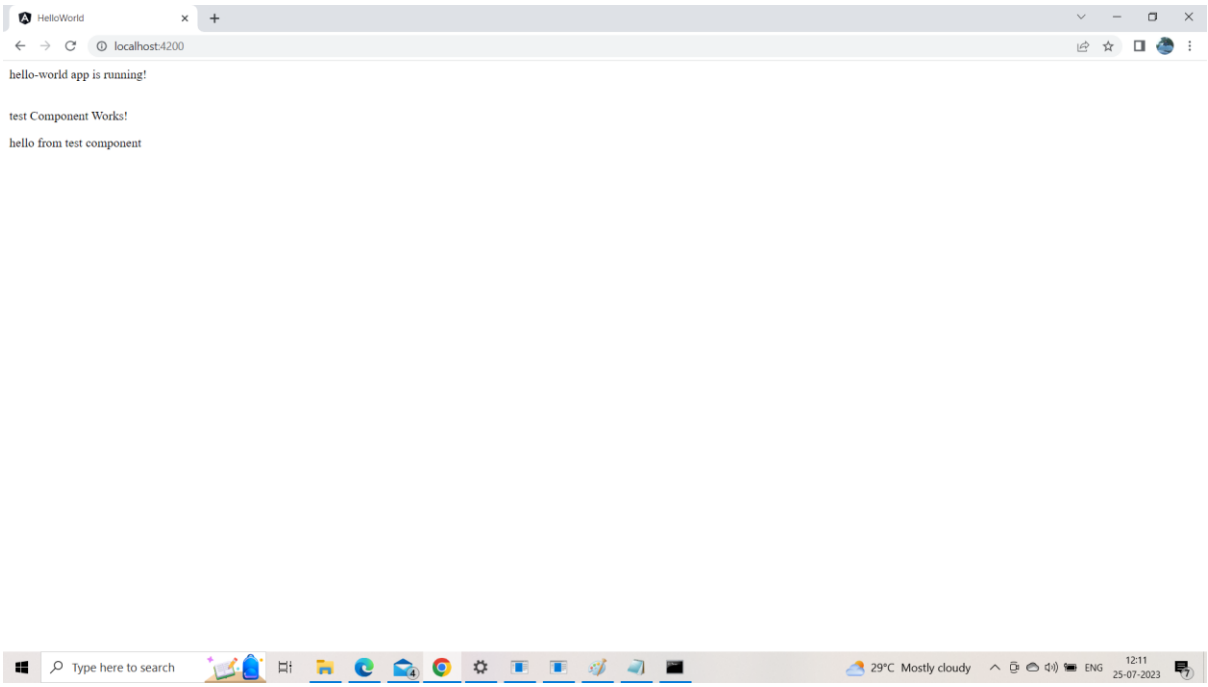
** **

**
**

```
  <div>
  </div>
 <app-test></app-test>
<router-outlet></router-outlet>
```

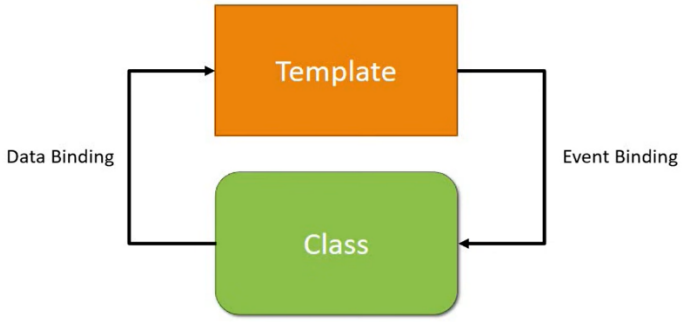
5. Run the application using below command

E:\Angular\hello-world>npm start



1 c . Add an event to the hello component template and when it is clicked, it should change the courseName.

What is event binding



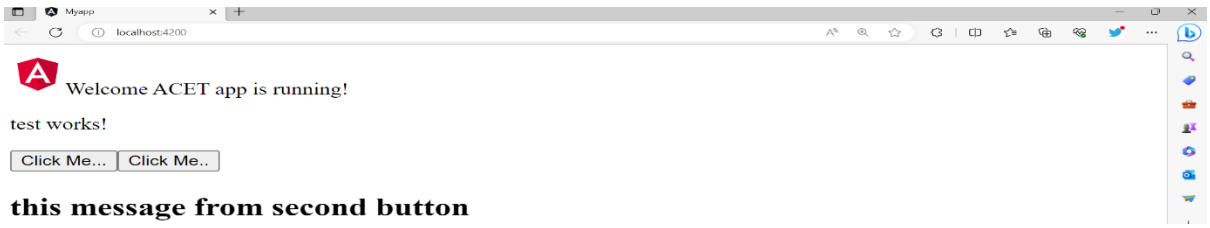
Step on go to test.component .html then create button like below

```
<p>test works!</p>
<button (click)="onclick()"> Click Me... </button>
<button (click)="message='this message from second button'"> Click Me..
</button>
<h2>{{message}} </h2>
```

Then go to test.componet.ts write the function onclick

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

@Component({
  selector: 'app-test',
  templateUrl: './test.component.html',
  styleUrls: ['./test.component.css']
})
export class TestComponent {
       //public name ="abc";
         public message = "";
       onclick()
       {
         this.message="this is my message..."
       }
}
```



2 a . Course Name : Structural Directives

Structural directives use for add or remove html elements

- ➔ **Nglf**
- ➔ **Ngswitch**
- ➔ **Ngfor**

Create a login form with username and password fields. If the user enters the correct credentials, it should render a "Welcome <>" message otherwise it should render "Invalid Login!!! Please try again..." message

test.component.html

```
<div *ngIf="!submitted">
    <form>
      <label>User Name</label>
      <input type="text" #username /><br /><br />
      <label for="password">Password</label>
      <input type="password" name="password" #password /><br />
    </form>
    <button (click)="onsubmit(username.value, password.value)">Login</button>
  </div>
  <div *ngIf="submitted">
    <div *ngIf="isAuthenticated; else failureMsg">
      <h4>Welcome {{ username }}</h4>
    </div>
    <ng-template #failureMsg>
      <h4>Invalid Login !!! Please try again...</h4>
    </ng-template>
    <button type="button" (click)="submitted = false">Back</button>
  </div>
```

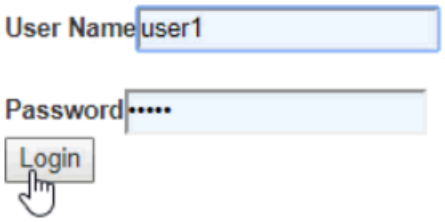
test.component.ts

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

@Component({
  selector: 'app-test',
  templateUrl: './test.component.html',
  styleUrls: ['./test.component.css']
})
export class TestComponent {
       //public name ="abc";
       isAuthenticated! : boolean;
       submitted = false;
       username! : string;

       onsubmit(name: string ,password: string)
```

```
        {
                this.submitted=true;
                this.username=name;
                if(name=='admin' && password=='admin')
                {
                   this.isAuthenticated=true;
                }
                else
                {
                  this.isAuthenticated=false;
                }
        }
}
```



ngFor:

ngFor directive is used to iterate over collection of data

2bCreate a courses array and rendering it in the template using ngFor directive in a list format

.html

```
<p>--- ngfor ---</p>
 <ul>
    <li *ngFor="let course of courses;  let i = index">
         {{i}} - {{ course}}
    </li>
  </ul>

  <ul>
    <li *ngFor="let subject of subjects">
       {{subject}}
    </li>
  </ul>
  <h1> names ... </h1>
  <ul>
    <li *ngFor="let name of names">
      {{name}}
    </li>
  </ul>
```

Output :

.ts

```
export class DirComponent {
   //public directives = "ngif||ngswitch||ngfor";
   displaymessage = true;
 //ng for
 courses: any[]=["Type script","Java Script","Node Js"];
 subjects : any[]=["IOT","CC"];

}
```

--- ngfor ---

- 0 - Type script
- 1 - Java Script
- 2 - Node Js

- IOT
- CC

ngSwitch

ngSwitch adds or remove DOM tree when their expression match the switch expression

2c) Display the correct option based on the value passed to ngSwitch directive.

.ts file

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

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

 choice=0;

 nextchoice()
 {
   this.choice++;
 }
}
```

`

.html

```
  <h2 class="title">Switch Case..</h2>

<div [ngSwitch]="choice">
<p *ngSwitchCase="1">{{choice}} First Choice </p>
<p *ngSwitchCase="2">{{choice}}  Second Choice </p>
<p *ngSwitchCase="3">{{choice}} Third Choice </p>
<p *ngSwitchDefault>{{choice}} Default Choice </p>
</div>
<button (click)="nextchoice()"> Next Choice </button>
```

Out put

2d) Create a custom structural directive called 'repeat' which should repeat the element given a number of times.

repeat.directive.ts

```
import { Directive, TemplateRef, ViewContainerRef,Input } from
'@angular/core';

@Directive({
  selector: '[appRepeat]'
})
export class RepeatDirective {
  constructor(private templateRef: TemplateRef<any>, private viewContainer:
ViewContainerRef) { }
  @Input() set appRepeat(count: number) {
    for (let i = 0; i < count; i++) {
      this.viewContainer.createEmbeddedView(this.templateRef);
    }
  }
}
```

app.component.html

```
<h2> repeat directive </h2>
 <p *appRepeat="5">hello</p>
```

Output:

Structural Directive

I am being repeated...

I am being repeated...

I am being repeated...

I am being repeated...

I am being repeated...

3a) Apply multipule css properties to a paragraph in a component using ngStyle

app.component.ts

```
export class AppComponent {
      title = "ACET";
      isactive = "Active";
      isBordered = true;   }
```

app.component.html

```
<p [ngStyle]="{color:isactive=='Active' ? 'green':'red'}"> Your Account is
{{isactive}}  </p>
```

Out put :

Your Account is Active

3b) Apply multiple css classes to the text using ngClass directive

app.component.html

```
<div [ngClass]="{bordered: isBordered}">
  Border {{ isBordered ? "ON" : "OFF" }}
</div>
```

app.componenet.ts

```
export class AppComponent {
      title = "ACET";
      isactive = "Active";
      isBordered = true;   }
```

3c) Module Name: Custom Attribute Directive

Module Name: Create an attribute directive called ‘Show Message’ which should display the given message in a paragraph when a user clicks on it and should change the text color to red.

Step1: create directive using below command

- ng generate directive 'ShowMessage'

(or)

- ng g d 'ShowMessage'

then we find two files with extension .ts and spec.ts

1.ShowMessage.directive.ts

2.ShowMessage.directive.spec.ts

Open ShowMessage.directive.ts the add below code

```
import { Directive,ElementRef,Renderer2,HostListener,Input} from
'@angular/core';

@Directive({
  selector: '[appShowmessage]'
})
export class ShowmessageDirective {
        @Input('appShowmessage') message!:string;
  constructor(private el: ElementRef,private render:Renderer2 )
  {
    render.setStyle(el.nativeElement,'cursor','pointer');

  }
  @HostListener('click') onClick(){
    this.el.nativeElement.innerHTML= this.message;
    this.render.setStyle(this.el.nativeElement,'color','red');
  }

}
```

Now Open the app.component.html then add below statement

```
<h3>College Information</h3>
<p [appShowmessage] = "myMessage">About Cse</p>
```

The run the application belwo command

- ng serve –open

College Information

About Cse

When we click the about cse then text will be change

Like below

College Information

240 Seats in computer science engineering..

4a. Module Name: Property Binding

Module Name: Property Binding

Binding image with class property using property binding

app.component.ts

```
export class AppComponent {

    imageurl ='assets/imgs/v.jpeg';
```

```
}
```

app.component.html

```
<img [src]="imageurl"/>
```

4b. Binding colspan attribute of a table element to the class property

app.component.ts

```
export class AppComponent {
    colspanvalue ="2";
  }
```

app.component.html

```
<table border="1" >
<tr>
  <td [attr.colspan]="colspanvalue"> CSE   </td>
  <td> IT </td></tr>
<tr>
  <td>ECE</td><td>EEE</td><td>MECH</td>
</tr>
</table>
```

Output:

CSE		IT
ECE	EEE	MECH

4c. Binding an element using inline style and user actions like entering text in input fields.

app.component.ts

```
export class AppComponent {

    isvalid=true;

  }
```

app.component.html

```
<button [style.color]="isvalid ? 'blue' : 'red' "> click </button>
<p [style.font-size.px]="isvalid ? 12 : 14"> font sie </p>
```

Output:

click

font size

5a. Display the product code in lowercase and product name in uppercase using built-in pipes

app.component.ts

```
export class AppComponent {
    title = "Product details";
    prodcutcode="prod_001";
    prodcutname="Laptop";
  }
```

app.component.html

```
<h3> {{ title | titlecase}} </h3>
<table style="text-align:left">
<tr><th> Product Code </th>
<td> {{ productcode | lowercase }} </td></tr>
<tr><th> Product Name </th>
<td> {{ productname | uppercase }} </td></tr>
</table>
```

Output:

Product Details

Product Code	prod_001
Product Name	LAPTOP

5b.Passing Parameters to Pipes. Apply built-in pipes with parameters to display product details

app.component.ts

```
export class AppComponent {
  productCode = 'PROD_P001';
  productName = 'Apple MPTT2 MacBook Pro';
  productPrice = 217021;
  purchaseDate = '1/17/2018';
  productTax = '0.1';
  productRating = 4.92;
```

```
  }
```

app.component.html

```
<table style="text-align:left">
    <tr>
        <th> Product Code </th>
        <td> {{ productCode | slice:5:9 }} </td>
    </tr>
    <tr>
        <th> Product Name </th>
        <td> {{ productName | uppercase }} </td>
    </tr>
    <tr>
        <th> Product Price </th>
        <td> {{ productPrice | currency: 'INR':'symbol' }} </td>
    </tr>
    <tr>
        <th> Purchase Date </th>
        <td> {{ purchaseDate | date:'fullDate' | lowercase}} </td>
    </tr>
    <tr>
        <th> Product Tax </th>
        <td> {{ productTax | percent : '.2' }} </td>
    </tr>
    <tr>
        <th> Product Rating </th>
        <td>{{ productRating | number:'1.3-5'}} </td>
    </tr>
</table>
```

OutPut:

Product Details

Product Code	P001
Product Name	APPLE MPTT2 MACBOOK PRO
Product Price	₹217,021.00
Purchase Date	wednesday, january 17, 2018
Product Tax	10.00%
Product Rating	4.920

5c. Nested Components Basics

Load Course List Component in the root component when a user click on the view courses list button.

Step 1: create courselist component

E:\Angular\myapp>ng generate component courselist

CREATE src/app/courselist/courselist.component.html (25 bytes)

CREATE src/app/courselist/courselist.component.spec.ts (587 bytes)

CREATE src/app/courselist/courselist.component.ts (218 bytes)

CREATE src/app/courselist/courselist.component.css (0 bytes)

UPDATE src/app/app.module.ts (886 bytes)

Step2: Open courselist.component.ts

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

@Component({
  selector: 'app-courselist',
  templateUrl: './courselist.component.html',
  styleUrls: ['./courselist.component.css']
})
export class CourselistComponent {
  courses = [{courseid:1,coursename:'nodejs'},
            {courseid:2,coursename:'reactjs'}
];

}
```

Step3: Open courselist.component.html

```
<table border="1">
    <thead>
      <tr>
        <th>Course ID</th>
        <th>Course Name</th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let course of courses">
        <td>{{ course.courseid }}</td>
        <td>{{ course.coursename }}</td>
      </tr>
    </tbody>
  </table>
```

Step4:- Open app.component.ts

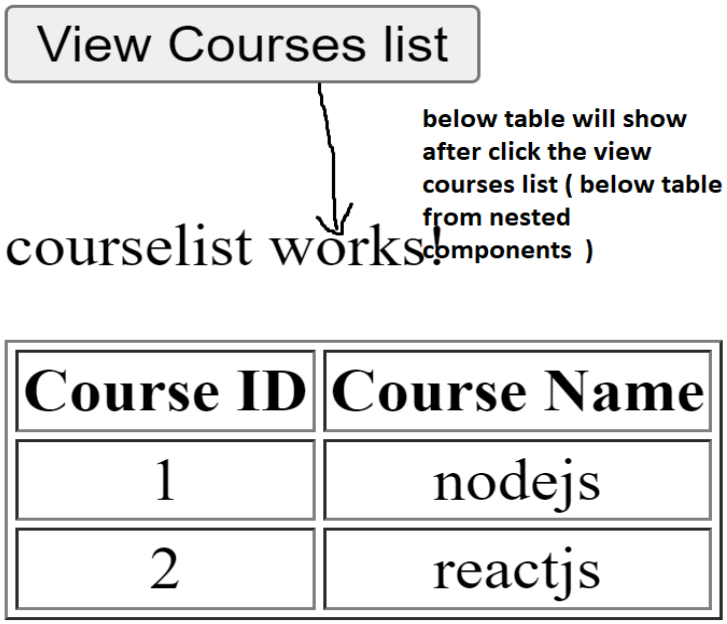
```
import { Component,OnInit } from '@angular/core';

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

Step5:- Open app.component.html

```
<button (click)="show = true">View Courses list</button><br /><br />
<div *ngIf="show">
<app-courselist></app-courselist>
  </div>
```

Step6:- run the application



6.a Create an APPComponent that displays a dropdown with a list of courses as values in it. Create another component called the coursesList component and load it in AppCOmponent which should display the course details . when the user selects a course .

Ans:

Already we create the courselist component

<u>Open courselist.component.ts add below code :</u>

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

```
@Component({
  selector: 'app-courselist',
  templateUrl: './courselist.component.html',
  styleUrls: ['./courselist.component.css']
})
export class CourselistComponent {
  courses = [{courseid:1,coursename:'NodeJS'},
            {courseid:2,coursename:'ReactJS'},
            {courseid:3,coursename:'AngularJS'}
            ];
            course!: any[];
            @Input() set cName(name: string) {
              this.course = [];
              for (var i = 0; i < this.courses.length; i++) {
                if (this.courses[i].coursename === name) {
                  this.course.push(this.courses[i]);
                }
              }
            }
}
```

Then open courselist.component.html and add below code

```
<p>courselist works!</p>

<table border="1" *ngIf="course.length > 0">
    <thead>
      <tr>
        <th>Course ID</th>
        <th>Course Name</th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let c of course">
        <td>{{ c.courseid }}</td>
        <td>{{ c.coursename }}</td>
      </tr>
    </tbody>
  </table>
```

Then open app.component.ts add below property

```
export class AppComponent {
name!: string;
  }
```

Then open app.component.html add below code

```
 Select a course to view
<select #course (change)="name = course.value">
```

```
  <option value="NodeJS">NodeJS</option>
  <option value="AngularJS">AngularJS</option>
  <option value="ReactJS">ReactJS</option></select><br /><br />
  <app-courselist [cName]="name"></app-courselist>
<router-outlet></router-outlet>
<app-test></app-test>
<app-dir></app-dir>
```