AngularJS is a JavaScript open-source front-end structural framework that is mainly used to develop single-page web applications (SPAs). It is a continuously growing and expanding framework which provides better ways for developing web applications. It changes the static HTML to dynamic HTML. Its features like dynamic binding and dependency injection eliminate the need for code that we have to write otherwise. AngularJS is rapidly growing and because of this reason, we have different versions of AngularJS with the latest stable being 1.7.9. It is also important to note that Angular is different from AngularJS. It is an open-source project which can be freely used and changed by anyone. It extends HTML attributes with Directives, and data is bound with HTML.

Why use AngularJS?

- **Easy to work with:** All you need to know to work with AngularJS is the basics of HTML, CSS, and JavaScript, not necessary to be an expert in these technologies.
- **Time-saving:** AngularJS allows us to work with components and hence we can use them again which saves time and unnecessary code.
- **Ready to use a template:** AngularJS is mainly plain HTML, and it mainly makes use of the plain HTML template and passes it to the DOM and then the AngularJS compiler. It traverses the templates and then they are ready to use.
- **Directives:** AngularJS's directives allow you to extend HTML with custom elements and attributes. This enables you to create reusable components and define custom behaviors for your application. Directives make it easier to manipulate the DOM, handle events, and encapsulate complex UI logic within a single component.

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

Step 1: Before install angular we need to install NodeJS and vs studio

Step 2: Open command prompt

Use these commands for install angular

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

  above command is used to install the angular

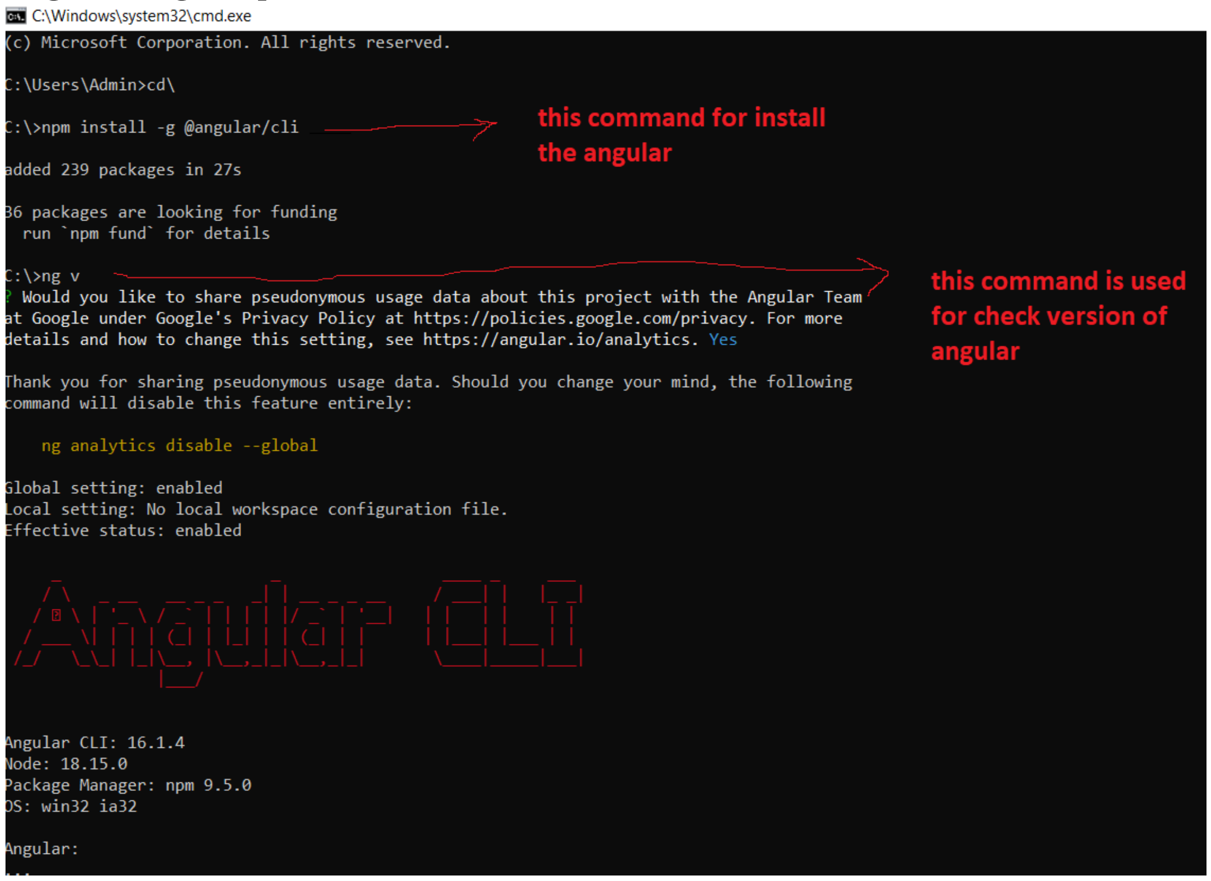
- **ng v**

  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



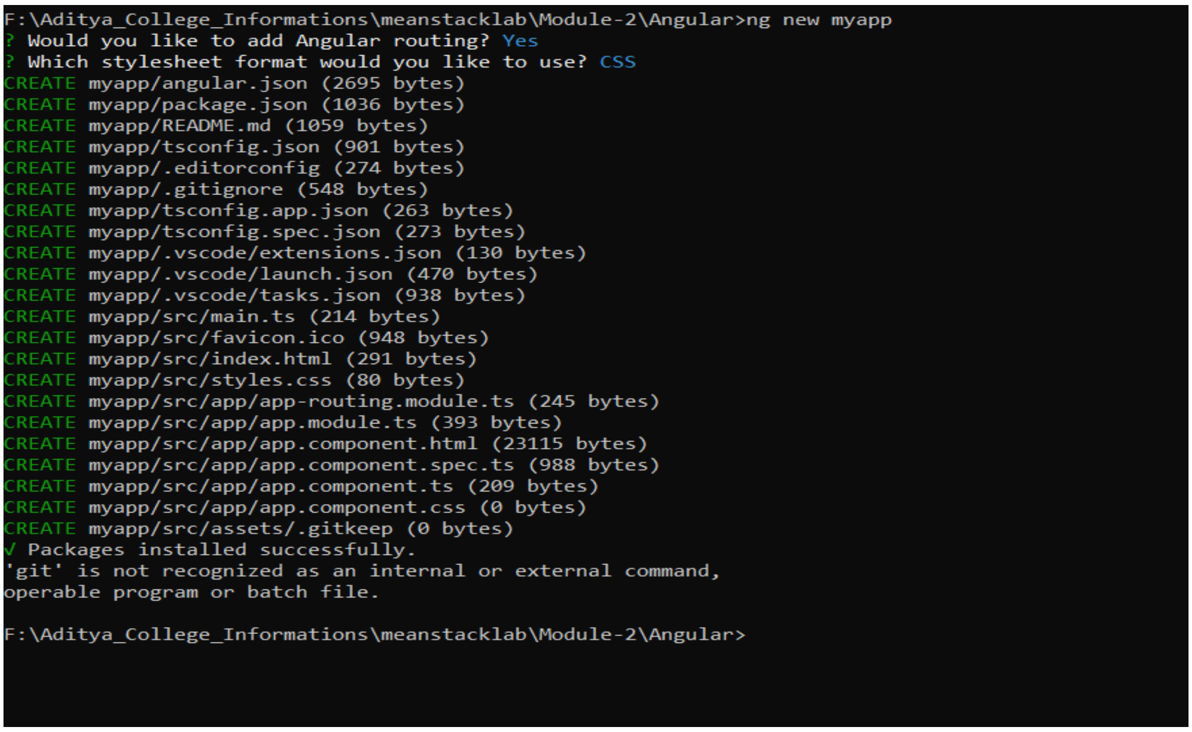
```
C:\Windows\system32\cmd.exe
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>cd\

C:\>npm install -g @angular/cli

added 239 packages in 27s

36 packages are looking for funding
  run `npm fund` for details

C:\>ng v
? Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.io/analytics. Yes

Thank you for sharing pseudonymous usage data. Should you change your mind, the following
command will disable this feature entirely:

    ng analytics disable --global

Global setting: enabled
Local setting: No local workspace configuration file.
Effective status: enabled

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

Angular:
```

this command for install the angular

this command is used for check version of angular

Creating a Angular Application

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



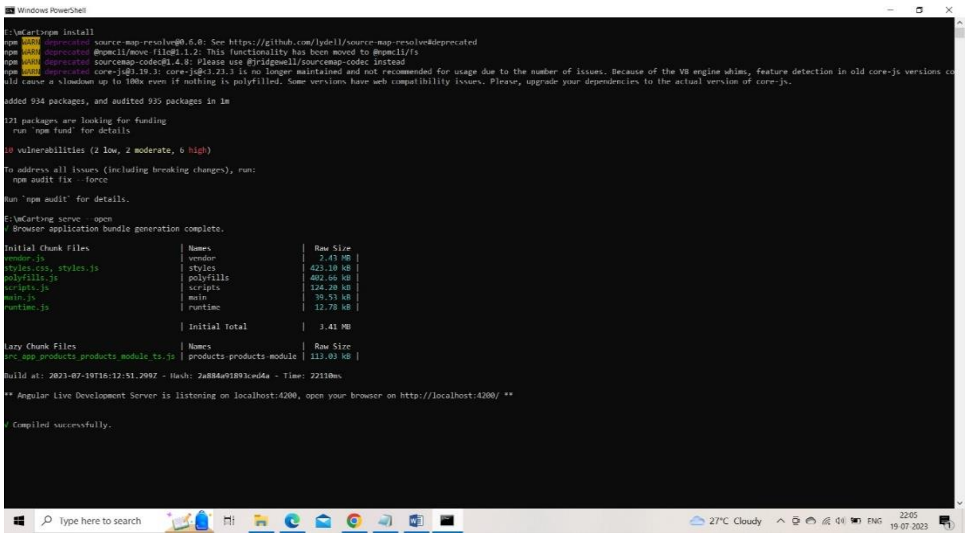
```
F:\Aditya_College_Informations\meanstacklab\Module-2\Angular>ng new myapp
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE myapp/angular.json (2695 bytes)
CREATE myapp/package.json (1036 bytes)
CREATE myapp/README.md (1059 bytes)
CREATE myapp/tsconfig.json (901 bytes)
CREATE myapp/.editorconfig (274 bytes)
CREATE myapp/.gitignore (548 bytes)
CREATE myapp/tsconfig.app.json (263 bytes)
CREATE myapp/tsconfig.spec.json (273 bytes)
CREATE myapp/.vscode/extensions.json (130 bytes)
CREATE myapp/.vscode/launch.json (470 bytes)
CREATE myapp/.vscode/tasks.json (938 bytes)
CREATE myapp/src/main.ts (214 bytes)
CREATE myapp/src/favicon.ico (948 bytes)
CREATE myapp/src/index.html (291 bytes)
CREATE myapp/src/styles.css (80 bytes)
CREATE myapp/src/app/app-routing.module.ts (245 bytes)
CREATE myapp/src/app/app.module.ts (393 bytes)
CREATE myapp/src/app/app.component.html (23115 bytes)
CREATE myapp/src/app/app.component.spec.ts (988 bytes)
CREATE myapp/src/app/app.component.ts (209 bytes)
CREATE myapp/src/app/app.component.css (0 bytes)
CREATE myapp/src/assets/.gitkeep (0 bytes)
√ Packages installed successfully.
'git' is not recognized as an internal or external command,
operable program or batch file.

F:\Aditya_College_Informations\meanstacklab\Module-2\Angular>
```

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



```
E:\mCart>npm install
npm WARN deprecated source-map-resolve@0.6.0: See https://github.com/lydell/source-map-resolve#deprecated
npm WARN deprecated @npmcli/move-file@1.1.2: This functionality has been moved to @npmcli/fs
npm WARN deprecated sourcemap-codec@1.4.8: Please use @jridgewell/sourcemap-codec instead
npm WARN deprecated core-js@3.19.3: core-js@<3.23.3 is no longer maintained and not recommended for usage due to the number of issues. Because of the V8 engine whims, feature detection in old core-js versions co
uld cause a slowdown up to 100x even if nothing is polyfilled. Some versions have web compatibility issues. Please, upgrade your dependencies to the actual version of core-js.

added 934 packages, and audited 935 packages in 1m

121 packages are looking for funding
  run `npm fund` for details

10 vulnerabilities (2 low, 2 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

E:\mCart>ng serve --open
√ Browser application bundle generation complete.

Initial Chunk Files                  | Names                     |  Raw Size
vendor.js                            | vendor                    |   2.43 MB |
styles.css, styles.js                | styles                    | 423.10 kB |
polyfills.js                         | polyfills                 | 402.66 kB |
scripts.js                           | scripts                   | 124.20 kB |
main.js                              | main                      |  39.53 kB |
runtime.js                           | runtime                   |  12.78 kB |

                                     | Initial Total             |   3.41 MB

Lazy Chunk Files                     | Names                     |  Raw Size
src_app_products_products_module_ts.js | products-products-module | 113.03 kB |

Build at: 2023-07-19T16:12:51.299Z - Hash: 2a884a91893ced4a - Time: 22110ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **


√ Compiled successfully.
```

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

Module
Component
Services

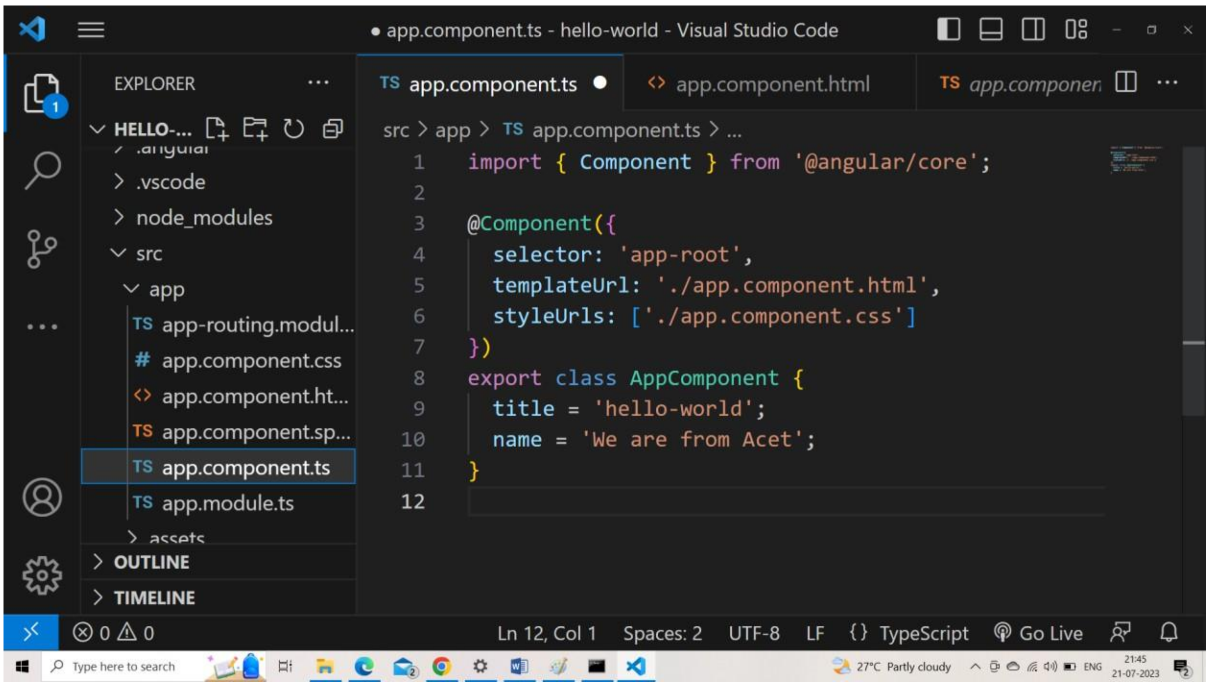
Let’s start the angular application is hello-world Create angular folder
E:\Angular>ng new hello-world
//Above command for create angular application E:\Angular>ng serve –open
//Execute angular application
Then open it visual studio go to src->app->app.component.ts Find title property add another called name
Then go to scr->app->app.component.html
Find the

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

Then add another tag with name property

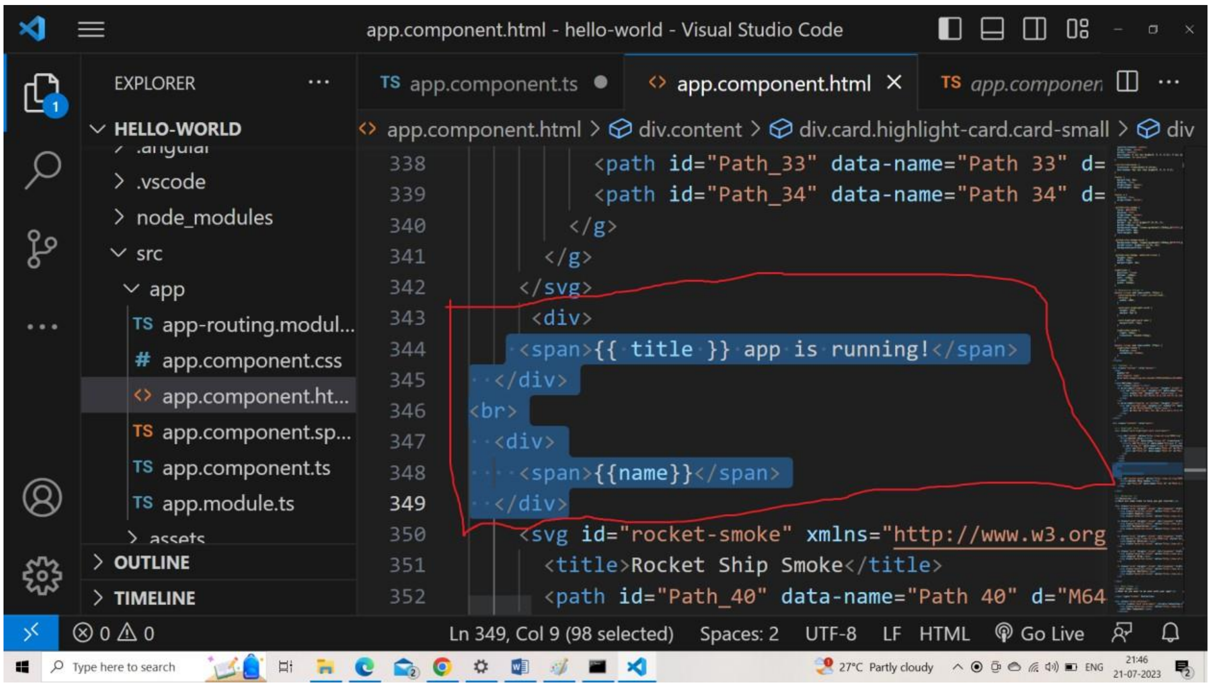
```
<div>
<span>{{name}}</span>
</div>
```

app.component.ts



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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'hello-world';
  name = 'We are from Acet';
}
```

app.component.html



app.component.html - hello-world - Visual Studio Code

EXPLORER

HELLO-WORLD
- .angular
- .vscode
- node_modules
- src
  - app
    - app-routing.modul...
    - app.component.css
    - app.component.ht...
    - app.component.sp...
    - app.component.ts
    - app.module.ts
  - assets

OUTLINE

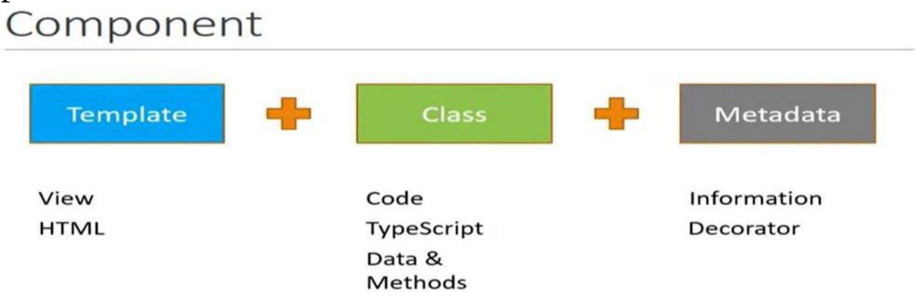
TIMELINE

app.component.html > div.content > div.card.highlight-card.card-small > div

```
            <path id="Path_33" data-name="Path 33" d=
            <path id="Path_34" data-name="Path 34" d=
          </g>
        </g>
      </svg>
      <div>
      <span>{{ title }} app is running!</span>
    </div>
    <br>
    <div>
      <span>{{name}}</span>
    </div>
      <svg id="rocket-smoke" xmlns="http://www.w3.org
        <title>Rocket Ship Smoke</title>
        <path id="Path_40" data-name="Path 40" d="M64
```

Ln 349, Col 9 (98 selected)   Spaces: 2   UTF-8   LF   HTML   Go Live

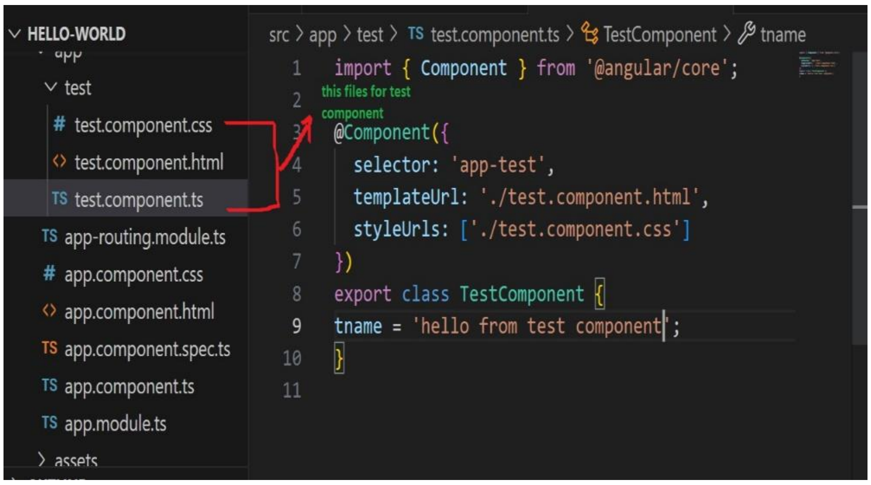
What is Component?



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



HELLO-WORLD
- app
  - test
    - test.component.css
    - test.component.html
    - test.component.ts
  - app-routing.module.ts
  - app.component.css
  - app.component.html
  - app.component.spec.ts
  - app.component.ts
  - app.module.ts
- assets

src > app > test > test.component.ts > TestComponent > tname

this files for test component

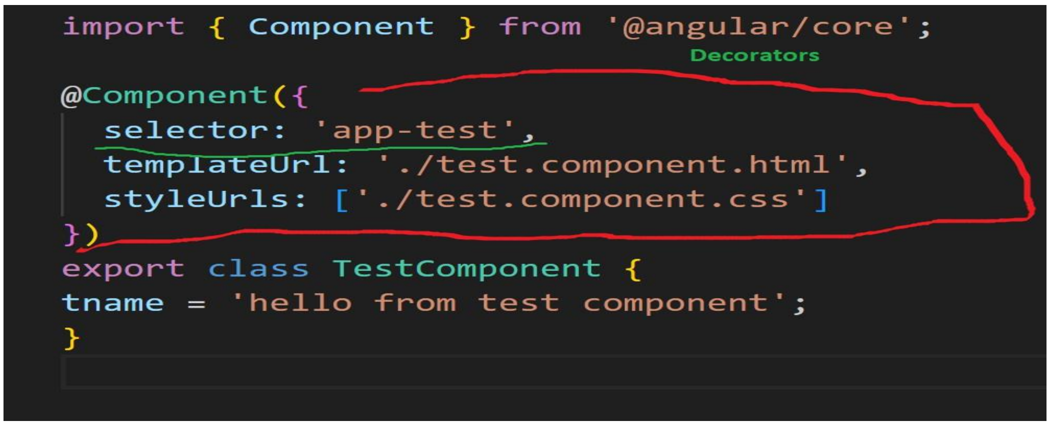
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-test',
  templateUrl: './test.component.html',
  styleUrls: ['./test.component.css']
})
export class TestComponent {
tname = 'hello from 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

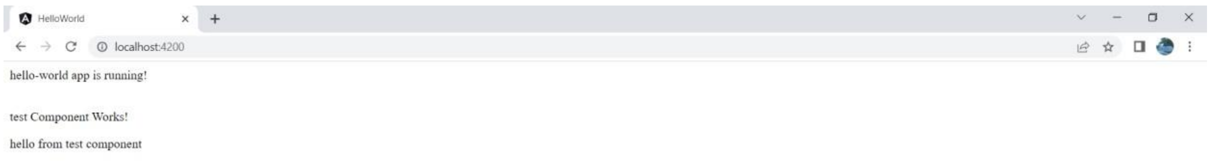


```
import { Component } from '@angular/core';

@Component({
  selector: 'app-test',
  templateUrl: './test.component.html',
  styleUrls: ['./test.component.css']
})
export class TestComponent {
tname = 'hello from test component';
}
```

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

```
<div>
<span>{{ title }} app is running!</span>
</div>
<br>
<div>
</div>
<app-test></app-test>
<router-outlet></router-outlet>
```

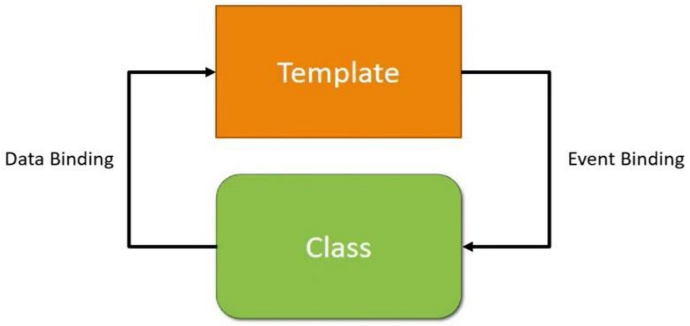
5. Run the application using below command E:\Angular\hello-world>**npm start**



```
hello-world app is running!

test Component Works!

hello from test component
```

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

What is event binding

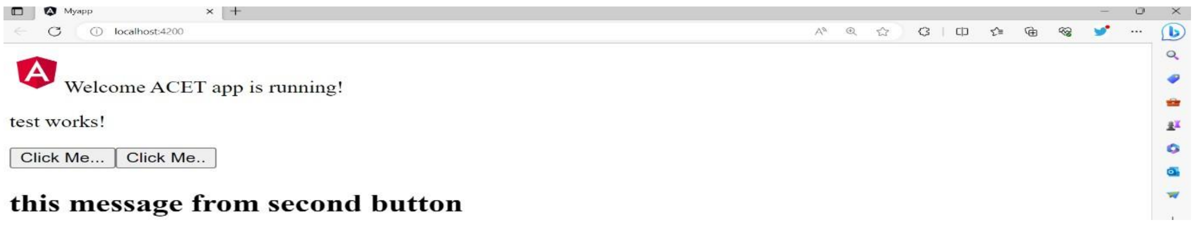


Step on go to **test.component.html** then create button like below

```
<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.component.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..."
       }
}
```



Welcome ACET app is running!

test works!

Click Me... Click Me..

this message from second button

2A. 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

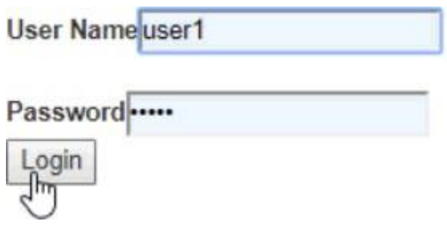
test.component.html

```
    <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>
```

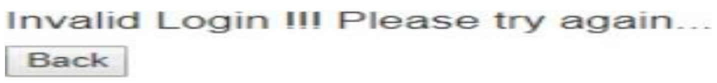
test.component.ts

```
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;
                }
        }
}
```


User Name user1

Password •••••


Invalid Login !!! Please try again...

ngFor:
ngFor directive is used to iterate over collection of data

2B. Create a courses array and rendering it in the template using ngFor directive in a list format

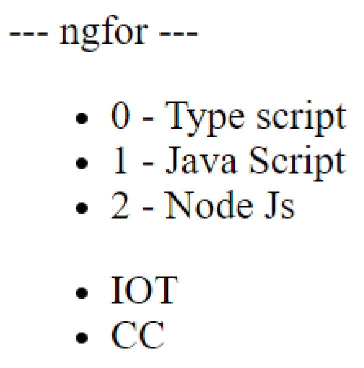
app.component.html

```
 <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>
```

app.component.ts

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

  }
```

Output:



```
--- ngfor ---

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

  • IOT
  • CC
```

<u>ngSwitch</u>

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.

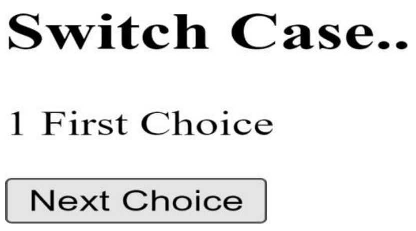
.ts file

```
     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++;
      }
     }
```

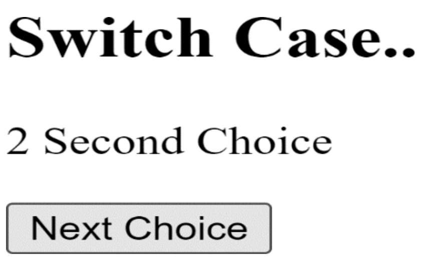
.html

```
  <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>
```

Output:



```
Switch Case..

1 First Choice
```



```
Switch Case..

2 Second Choice
```

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

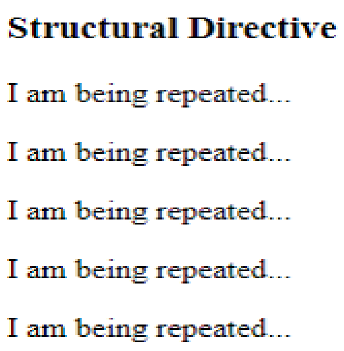
repeat.directive.ts

```
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 multiple css properties to a paragraph in a component using ngStyle

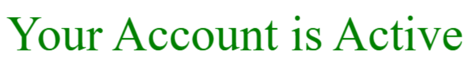
app.component.ts

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

app.component.html

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

Output:



```
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

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

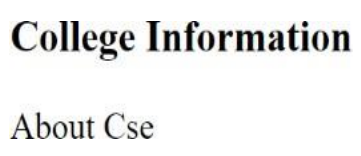
```
import { Directive,ElementRef,Renderer2,HostListener,Input} from
'@angular/core';

@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 below command

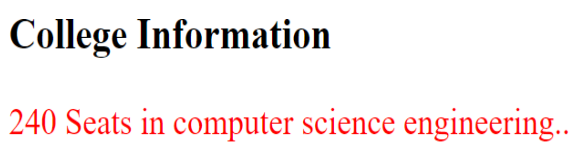
- **ng serve –open**



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

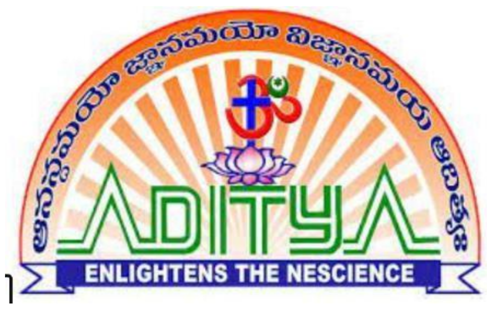
app.component.ts

```
export class AppComponent {
    imageurl ='assets/imgs/v.jpeg';
}
```

app.component.html

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

Output:



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

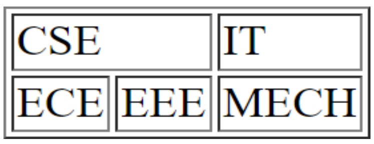
app.component.ts

```
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.

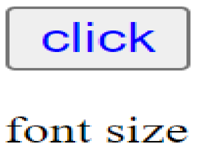
app.component.ts

```
 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:



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

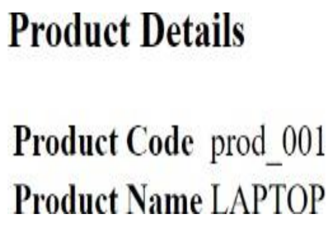
app.component.ts

```
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> {{ prodcutcode | lowercase }} </td></tr>
<tr><th> Product Name </th>
<td> {{ prodcutname | uppercase }} </td></tr>
</table>
```

Output:



```
Product Details

Product Code  prod_001
Product Name LAPTOP
```

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

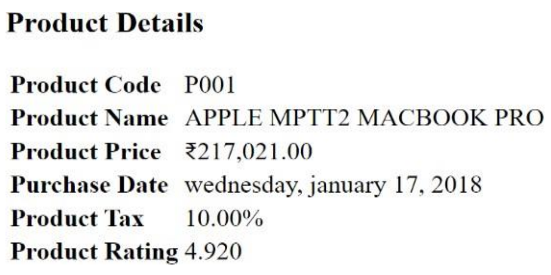
app.component.ts

```
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**

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>
```

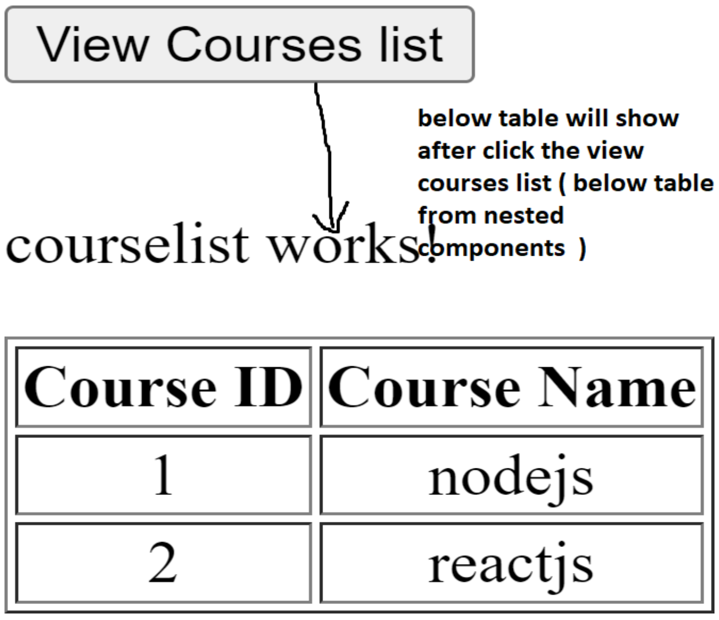
Step4:- Open app.component.ts

```
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



View Courses list
below table will show after click the view courses list (below table from nested components)

courselist works!

Course ID	Course Name
1	nodejs
2	reactjs

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]);
                }
              }
            }
}
```

<u>Then open courselist.component.html and add below code</u>

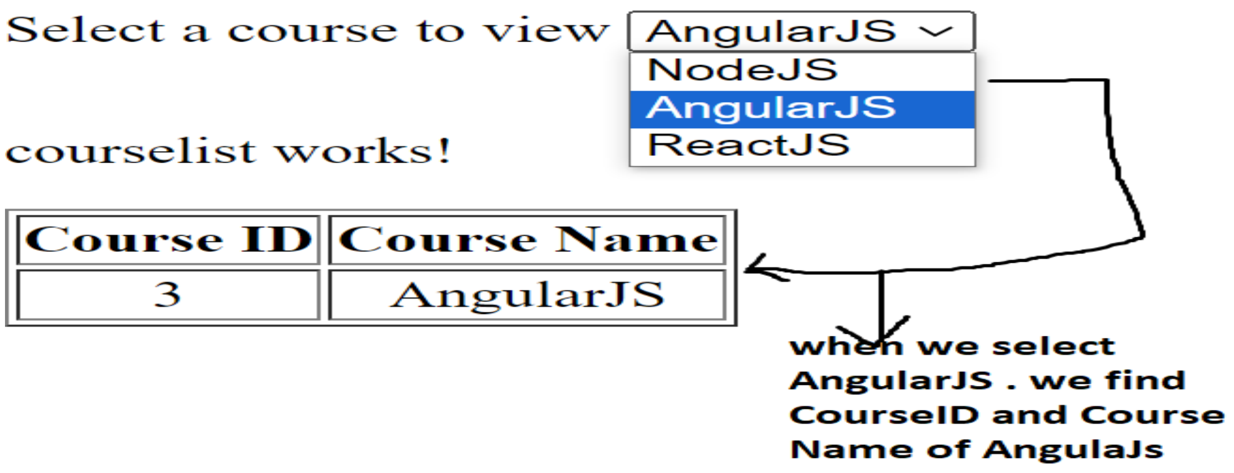
```
<p>courselist works!</p>

<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>
```



Select a course to view

courselist works!

Course ID	Course Name
3	AngularJS

when we select AngularJS . we find CourseID and Course Name of AngulaJs

6b) Passing Data from child component to container component Create an AppComponent that loads another component called the course List Component. Create another component called course list component which should display the courses list in a table along with a register.

Step1: create component course-list

Then open course-list.ts then add below code

```
import { Component, OnInit, Input, Output, EventEmitter } from '@angular/core';
@Component({
  selector: 'app-courses-list',
  templateUrl: './courses-list.component.html',
  styleUrls: ['./courses-list.component.css']
})
export class CoursesListComponent {
  @Output() registerEvent = new EventEmitter<string>();
  courses = [
    { courseId: 1, courseName: 'Node JS' },
    { courseId: 2, courseName: 'Typescript' },
    { courseId: 3, courseName: 'Angular' },
    { courseId: 4, courseName: 'React JS' }
  ];
  register(courseName: string) {
    this.registerEvent.emit(courseName);
  }
}
```

Step2: now open course-list.component.html

```
<table border="1">
  <thead>
    <tr>
      <th>Course ID</th>
      <th>Course Name</th>
      <th></th>
    </tr>
  </thead>
  <tbody><tr *ngFor="let course of courses">
      <td>{{ course.courseId }}</td>
      <td>{{ course.courseName }}</td>
      <td><button (click)="register(course.courseName)">Register</button></td>
    </tr>
  </tbody>
</table>
```

Step3: Then open App.component.html then add below code

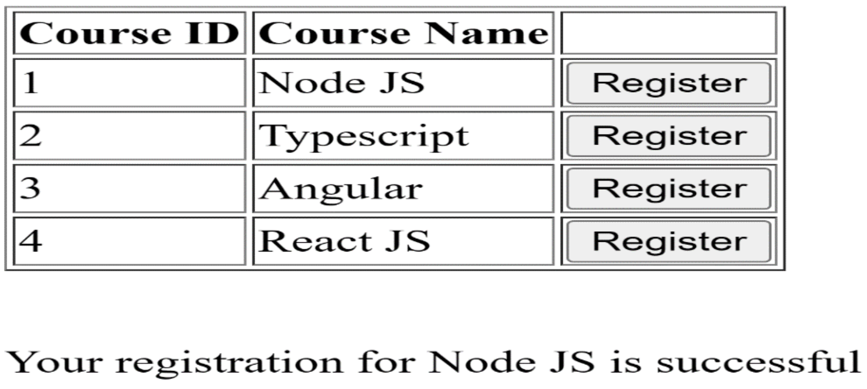
```
 <h2>Courses List</h2>
<app-courses-list (registerEvent)="courseReg($event)"></app-courses-list>
<br /><br />
<div *ngIf="message">{{ message }}</div>
```

Step4: Then open App.component.ts then add below code

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

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  message!: string;
  courseReg(courseName: string) {
    this.message = `Your registration for ${courseName} is successful`;
  }
}
```

Output:



Course ID	Course Name	
1	Node JS	Register
2	Typescript	Register
3	Angular	Register
4	React JS	Register

Your registration for Node JS is successful

6c) Apply Shadow DOM and Node encapsulation modes to component

Create Component Name called Component1. Using below command

- ng g c component1

Open component1.css

```
.cmp {
    padding: 6px;
    margin: 6px;
    border: blue 2px solid;
  }

Opent Component1.html
<p>component1 works!</p>
<div class="cmp">First Component</div>
```

Create Component Name called Component2. Using below command

- ng g c component1

Open component2.css

```
.cmp {
    padding: 6px;
    margin: 6px;
    border: blue 2px solid;
  }
```

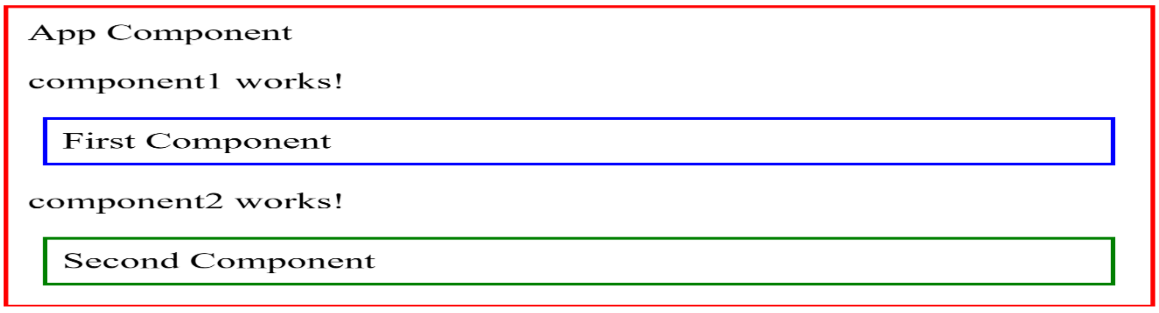
Opent Component1.html

```
<p>component1 works!</p>
<div class="cmp">Second Component</div>
```

Now open the appcomponent.html add below script

```
<div class="cmp">
    App Component
    <app-component1></app-component1>
    <app-component2></app-component2>
</div>
```

Output:



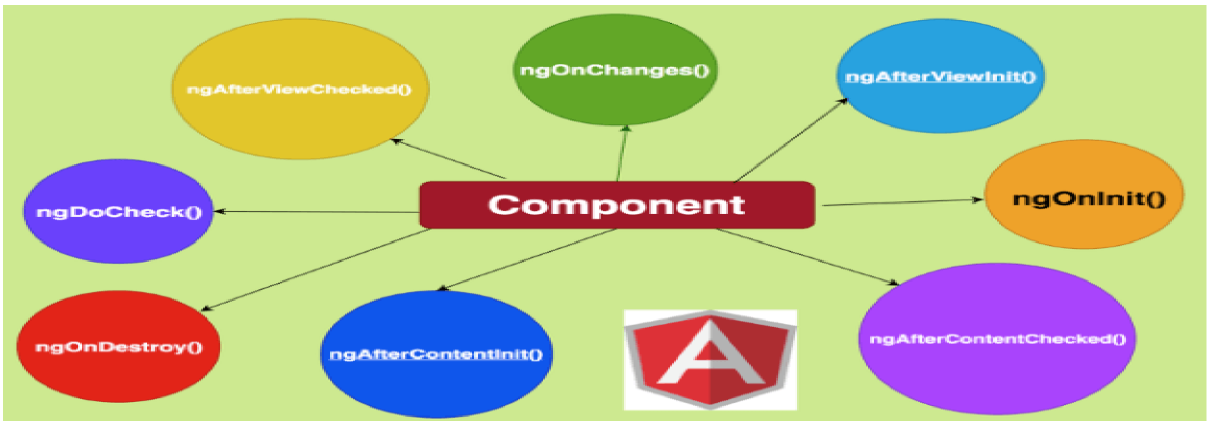
```
App Component
component1 works!
First Component
component2 works!
Second Component
```

6d. Override component life cycle hooks and logging the corresponding message to understand the flow

Component life cycle:

What is shadow DOM:-
Shadow DOM is a feature of the Web Components standard that allows you to create encapsulated DOM trees. This means that you can create a DOM tree that is separate from the main DOM tree of the document, and that styles and scripts applied to the main DOM tree will not affect the shadow DOM tree.
Component Life Cycle:
Every component has a life-cycle, a number of different stages it goes through from Initializing to destroying. There are 8 different stages in the component lifecycle. Every stage is called life cycle hook events so we can use these hook events in different phases of our applications to obtains fine controls on the components.
Since a component is a typescript class, for that reason every component must have a constructor method.

Interface	Hook	Support
OnChanges	ngOnChanges	Directive, Component
OnInit	ngOnInit	Directive, Component
DoCheck	ngDoCheck	Directive, Component
AfterContentInit	ngAfterContentInit	Component
AfterContentChecked	ngAfterContentChecked	Component
AfterViewInit	ngAfterViewInit	Component
AfterViewChecked	ngAfterViewChecked	Component
OnDestroy	ngOnDestroy	Directive, Component



Open app.component add below code :

```
import {
    Component, OnInit, DoCheck, AfterContentInit, AfterContentChecked,
    AfterViewInit, AfterViewChecked,
    OnDestroy
} from '@angular/core';
@Component({
    selector: 'app-root',
    styleUrls: ['./app.component.css'],
    templateUrl: './app.component.html'
})
export class AppComponent implements OnInit, DoCheck,
    AfterContentInit, AfterContentChecked,
    AfterViewInit, AfterViewChecked,
    OnDestroy {
    data = 'Angular';
    ngOnInit() {
        console.log('Init');
    }
    ngDoCheck(): void {
        console.log('Change detected');
    }
    ngAfterContentInit(): void {
        console.log('After content init');
    }
    ngAfterContentChecked(): void {
        console.log('After content checked');
    }
    ngAfterViewInit(): void {
        console.log('After view init');
    }
    ngAfterViewChecked(): void {
        console.log('After view checked');
    }
    ngOnDestroy(): void {
        console.log('Destroy');
    }
}
```

2. Write the below-given code in app.component.html

```
<div>
  <h1>I'm a container component</h1>
  <input type="text" [(ngModel)]="data" />
  <app-child [title]="data"></app-child>
</div>
```

3. Write the below-given code in child.component.ts

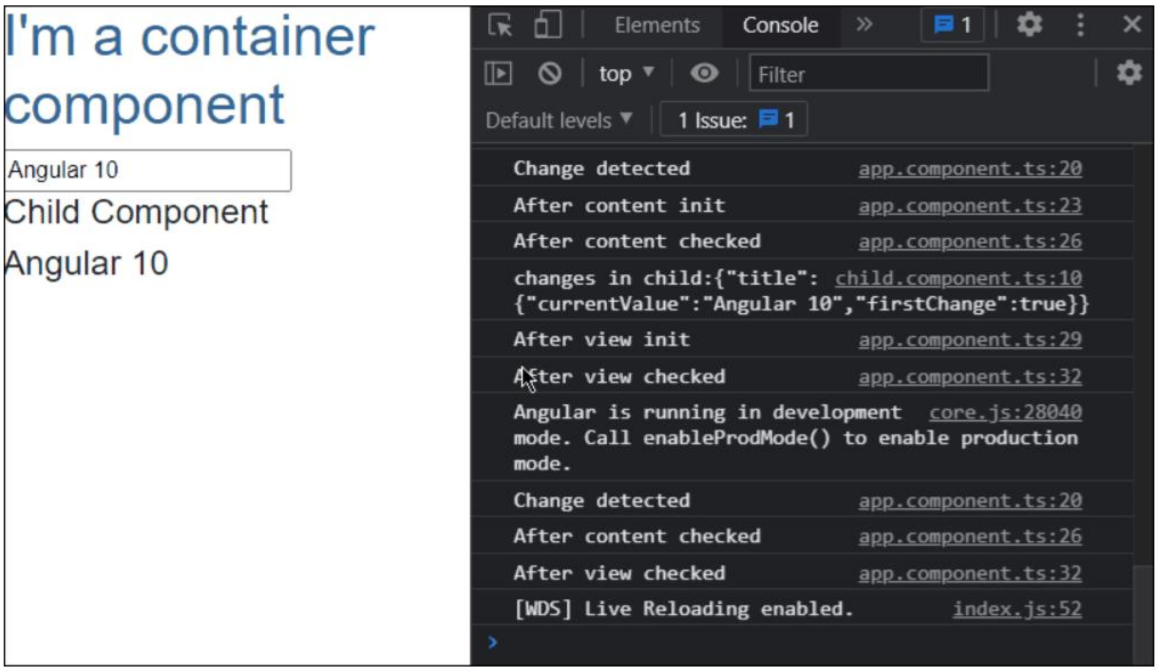
```
import { Component, OnChanges, Input } from '@angular/core';
@Component({
  selector: 'app-child',
  templateUrl: './child.component.html',
  styleUrls: ['./child.component.css']
})
export class ChildComponent implements OnChanges {
  @Input() title!: string;
  ngOnChanges(changes: any): void {
    console.log('changes in child:' + JSON.stringify(changes));
  }
}
```

4. Write the below-given code in child.component.html

```
<h2>Child Component</h2>
```

```
<h2>{{title}}</h2>
```

Output:



```
Change detected                                  app.component.ts:20
After content init                               app.component.ts:23
After content checked                            app.component.ts:26
changes in child:{"title":                     child.component.ts:10
{"currentValue":"Angular 10","firstChange":true}}
After view init                                  app.component.ts:29
After view checked                               app.component.ts:32
Angular is running in development                    core.js:28040
mode. Call enableProdMode() to enable production
mode.
Change detected                                  app.component.ts:20
After content checked                            app.component.ts:26
After view checked                               app.component.ts:32
[WDS] Live Reloading enabled.                          index.js:52
```