MALLA REDDY UNIVERSITY

School of Engineering

Computer Science Engineering

MEAN Stack Web Development Lab

Week 1: Installation of Software and Practice TypeScript

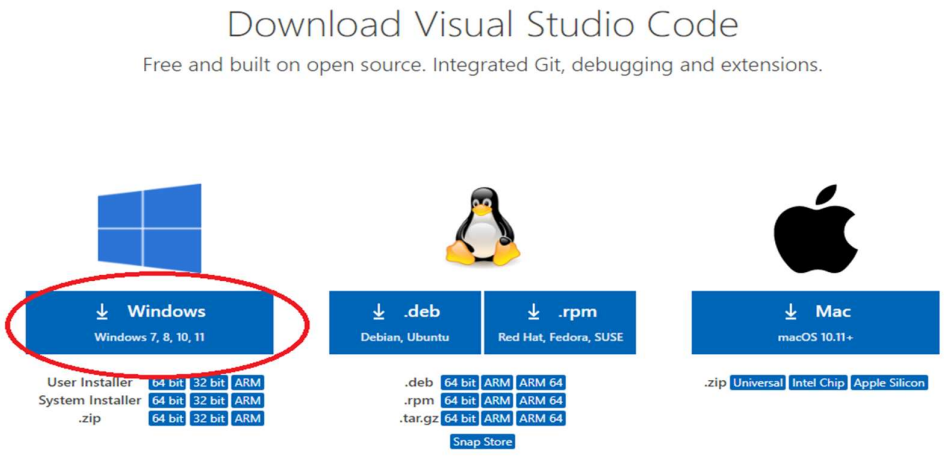
A. Installation of Software

1. Install Visual Studio
2. Install Node JS
    a. Check version of Node and NPM installed
3. Install TypeScript
    a. Check Version of TypeScript
    b. Get help of TypeScript
    c. Practice TypeScript

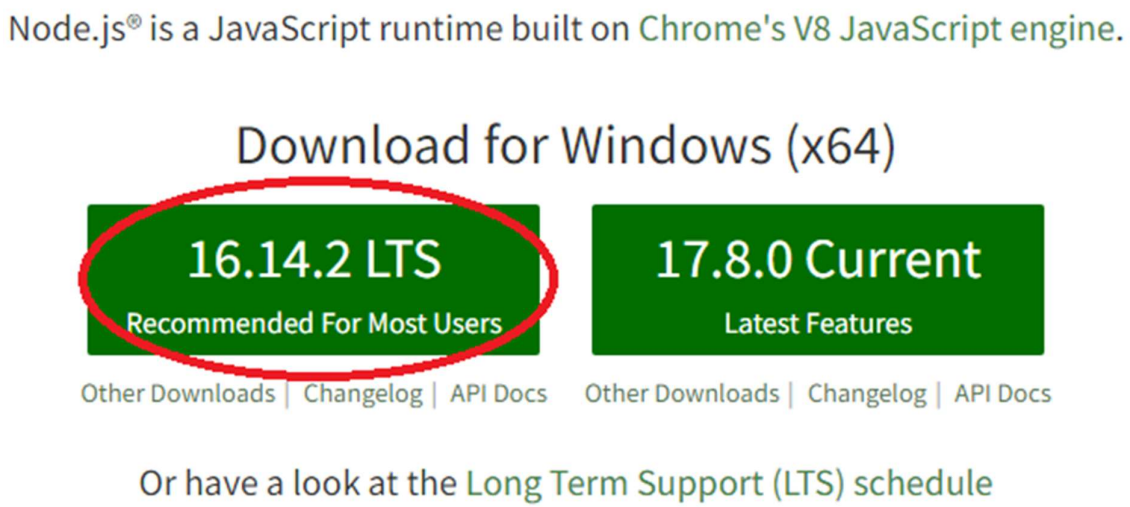
Procedure:

1. Install Visual Studio Code Software

Download Visual Studio Code - Mac, Linux, Windows



2. Install Node JS from https://nodejs.org



2a. Check version of Node and NPM installed

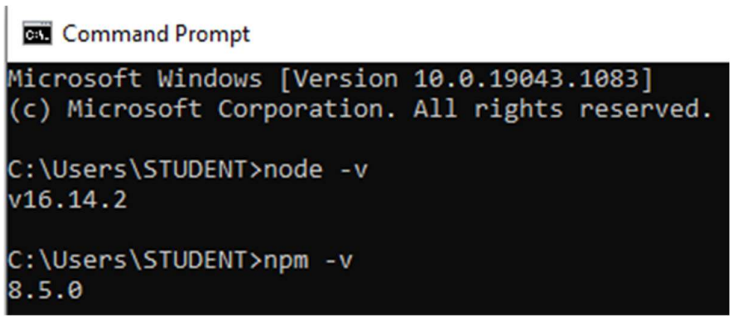
Go to command prompt

Verify Installed Node Version

Enter node -v

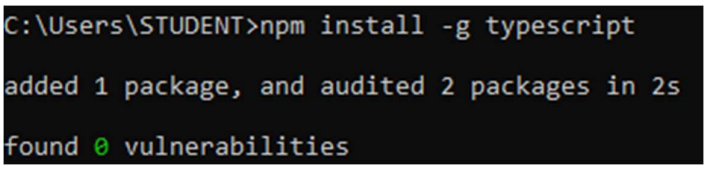
Verify Installed NPM Version

Enter npm -v



3. Install TypeScript

Enter npm install -g typescript



3a. Check Version of typescript

Enter tsc --version

```
C:\Users\STUDENT>tsc --version
Version 4.6.3
```

3b. Get help of typescript

Enter tsc --help

```
C:\Users\STUDENT>tsc --help
tsc: The TypeScript Compiler - Version 4.6.3

COMMON COMMANDS

  tsc
  Compiles the current project (tsconfig.json in the working directory.)

  tsc app.ts util.ts
  Ignoring tsconfig.json, compiles the specified files with default compiler options.

  tsc -b
  Build a composite project in the working directory.

  tsc --init
  Creates a tsconfig.json with the recommended settings in the working directory.

  tsc -p ./path/to/tsconfig.json
  Compiles the TypeScript project located at the specified path.
```

B. Practice Type Script

Procedure:

1. Open cmd.exe
2. Create Project Folder
3. Go to Project Folder and type code .
   OR
4. Open VS Code and Select Project Folder
5. Create Exercise.ts file
6. Convert Exercise.ts file to Exercise.js file using tsc Exercise.ts command
7. Run Exercise.js file using node Exercise.js command and check results

---

Primitive Types of TypeScript

```
// 1. Declaration of Number
let first: number=25;            // number
let second: number = 0x37CF;     // hexadecimal
let third: number = 0o377 ;      // octal
let fourth: number = 0b111001;   // binary
console.log("decimal value of 25: " + first)        // 123
console.log("hexadecimal value of 25: " + second);  // 14287
console.log("octal value of 25: " + third);         // 255
console.log("binary value of 25: " + fourth);
```

Write JS file:

Output:

```
// 2. Declaration of String
let str:string = "MALLAREDDY UNIVERSITY"
console.log("string is: "+str);
```

Write JS file:

Output:

```
//3. Declaration of Any
let val1: any = 'Hi';
console.log("value using any: " + val1) ;

val1 = 555;   // OK
console.log("value using any: " + val1) ;

val1 = true;   // OK
console.log("value using any: " + val1) ;
```

Write JS file:

Output:

```
//4. Declaration of Boolean
let isWorking:boolean=true;
console.log("boolean value: " + isWorking);

//5. Declaration of Union Type
let code:(string | number)
code = 123;
code = "abc";
console.log(code);
```

Write JS file:

Output:

Week 2: Practice TypeScript – Object Types

```
//1. Declaration of Array
let a : number[] = [1, 3, 5];
let b : Array<number> = [1, 3, 5];
let c :number[]=new Array(10,20,30,);
console.log("array1: "+ a);
console.log("array2: "+ b);
console.log("array3: "+ c);
```

Output:

```
//2. Declaration of Enum
enum directions {
    up = 1,
    down = 2,
    right = 3,
    left = 4
}
console.log(directions.up);
console.log(directions.down);
console.log(directions.right);
console.log(directions.left);
```

Output:

```
//3. Declaration of Object
let emp:[number,string] = [1,"steve Jobs"];
console.log(emp);
```

Output:

```
//4. Declaration of Class
class Student {
    rollNumber: number;
    studentName: string;

    constructor(rollNumber: number, studentName: string) {
        this.rollNumber = rollNumber;
        this.studentName=studentName;
    }

    display() {
      console.log(this.rollNumber);
      console.log(this.studentName);
    }
 }

 let studentObject = new Student(100,"Steve");
 studentObject.display();

//5. Declaration of Interface
interface Person
 {
    firstName: string;
    lastName: string;
    age: number;

    FullName();  //function
    GetAge();    //function
}

//5a. implementing the interface
class Employee implements Person {
    firstName: string;
    lastName: string;
    age:number;

    FullName() {
        return this.firstName + ' ' + this.lastName;
    }

    GetAge() {
        return this.age;
    }

    constructor(firstN: string, lastN: string, getAge: number) {
        this.firstName = firstN;
        this.lastName = lastN;
        this.age = getAge;
    }
```

```
}

//5b. using the class that implements interface
let myEmployee = new Employee('aaa', 'bbb', 25);
let fullName = myEmployee.FullName();
let Age = myEmployee.GetAge();
console.log("Name of Person: " + fullName + '\nAge: ' + Age);
```

Output:

```
//6. implementation of Module
// Employeemodule.ts:
export let age : number = 20;
export class Employee {
    empCode: number;
    empName: string;
    constructor(name: string, code: number) {
        this.empName = name;
        this.empCode = code;
    }
    displayEmployee() {
        console.log ("Employee Code: " + this.empCode + ", Employee Name: " +
this.empName );
    }
}
let companyName:string = "XYZ";

// Employee1.ts
import { Employee } from "./Employeemodule";

let empObj = new Employee("Rani", 1);
empObj.displayEmployee();
```

Output:

Week 3: Creation of Angular Project

Procedure:

1. Install Angular CLI
    - Check Version of Angular
2. Create Angular Project
3. Install Bootstrap and Configure Bootstrap
4. Open Project in Visual Studio Code and Build Project
5. Run Project

Note:

If you are unable to run from Terminal and getting error **"PS1 Can Not Be Loaded Because Running Scripts Is Disabled On This System In Angular"**, run the following commands on Terminal.

1. set-ExecutionPolicy RemoteSigned -Scope CurrentUser
2. Get-ExecutionPolicy
3. Get-ExecutionPolicy –list

Type Command ng –version to verify Terminal is now working or not.

If you are unable Create Angular Project, run the following command on Terminal.

npm cache clear --force

Procedure:

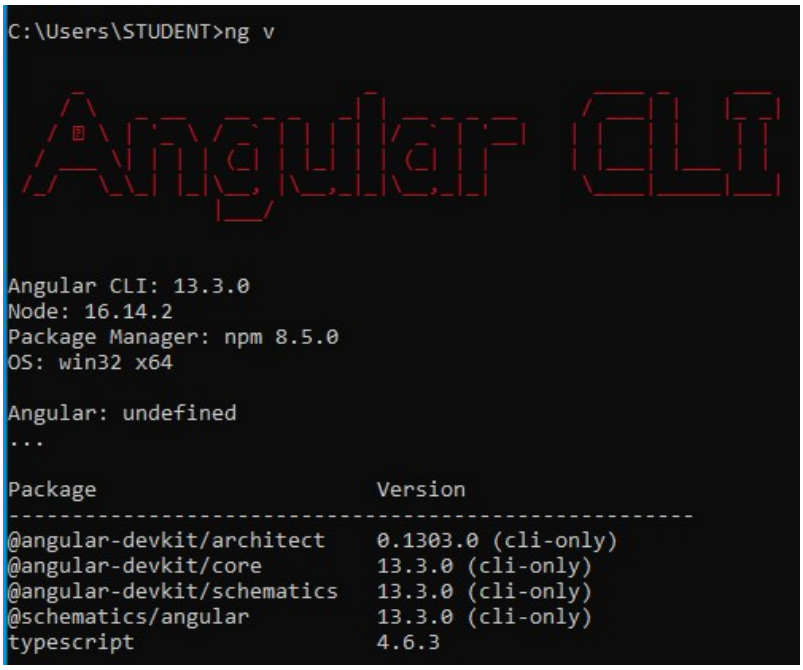
1. Install Angular CLI

Install the CLI using the npm package manager:

npm install –g @angular/cli

1a. Check version

ng v



2. Create Angular Project and Run Project

Go to Folder and Enter ng new [project name]

3. Install Bootstrap

npm install bootstrap -save

Configure Bootstrap 5 into Angular App

And navigate to your project and open angular.json file. And then add the following code into it; as follows:

```
"styles": [
            ...
            "node_modules/bootstrap/dist/css/bootstrap.min.css"
        ],
"scripts": [
            ...
            "node_modules/bootstrap/dist/js/bootstrap.min.js"
        ]
```

4. Open Project Folder in Visual Studio Code and Write

<h1>Hello World</h1>

in app.component.html file.

5. Run Project

Open Terminal and type below command

ng serve -o

You will find the below result on Default Browser at localhost:4200

Hello World

Then Apply Bootstrap class to check whether it is working or not

<h1 class='text-primary'>Hello World</h1>

If it works, you will see message as shown below on Browser

Hello World

Week 4: Creation of Angular Components

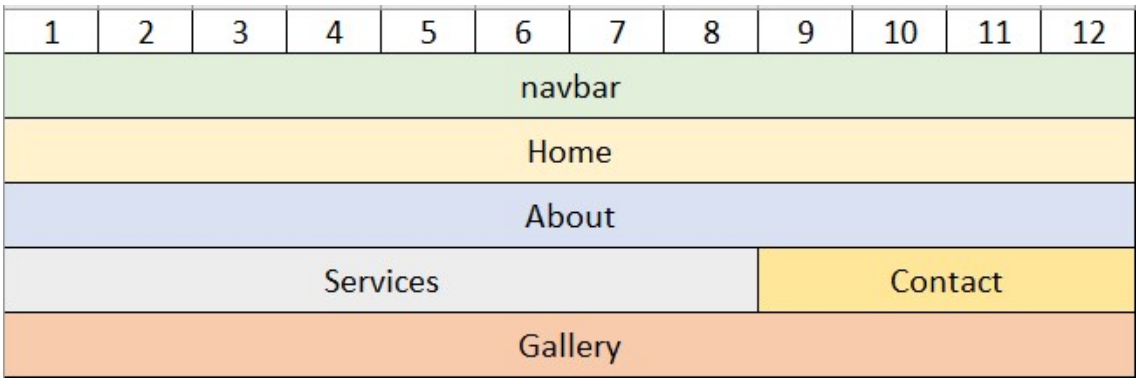
Objective:

To create a Website with Custom Components

Procedure:

1. Create Angular Project
2. Install Bootstrap and Configure Bootstrap
3. Open Project in Visual Studio Code
4. Create Custom Components: home, about, services, contact, gallery
5. Add Custom Components to app.component.html file
6. Run Project

Website Template



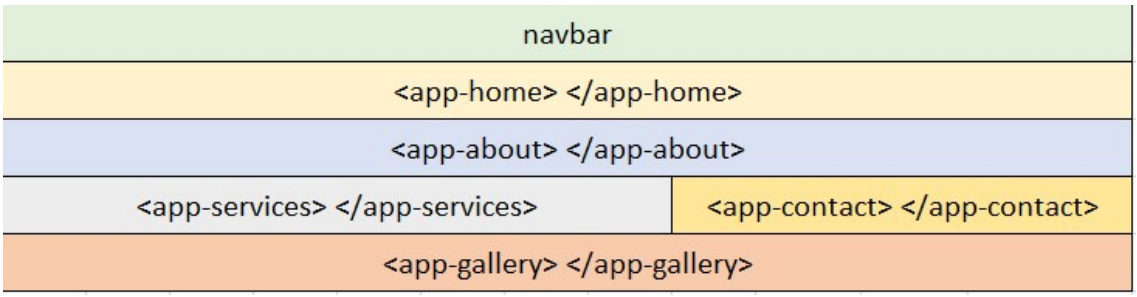
Step 4: Create Custom Components

ng g c [component name]

Use the following commands to create required Angular Components:

ng g c home → check home.component.ts for selector: app-home
ng g c about → check about.component.ts for selector: app-about
ng g c services → check services.component.ts for selector: app-services
ng g c contact → check contact.component.ts for selector: app-contact
ng g c gallery → check gallery.component.ts for selector: app-gallery

5. Add Custom Components in app.component.html file




```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container">
    <a class="navbar-brand" href="#">Single Page Application (SPA) - without Routing</a>
  </div>
</nav>
<div class="container border">
  <div class="row border-bottom">
    <div class="col-md-12"><app-home></app-home></div>
  </div>
  <div class="row border-bottom">
    <div class="col-md-12"><app-about></app-about></div>
  </div>
  <div class="row border-bottom">
    <div class="col-md-8"><app-services></app-services></div>
    <div class="col-md-4 border-start" ><app-contact></app-contact></div>
  </div>
  <div class="row border-bottom">
    <div class="col-md-12"><app-gallery></app-gallery></div>
  </div>
</div>
```

Step 6: Run the Project

Week 5: Single Page Application (SPA) with Routing

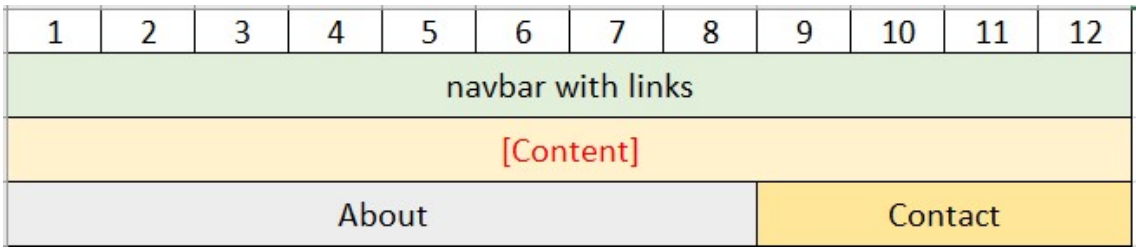
Objective:

To create a Single Page Application (SPA) Website with Custom Components using Routing.

Procedure:

Step 1: Create New Angular Project with Routing Option
Step 2: Install bootstrap and configure to the Project
Step 3: Create Components: home, services, about, contact and gallery
Step 4: Update app.component.html file with navbar, Router Outlet and other components
Step 5: Update app-routing.module.ts file
→ Import Components
→ Add Route Configuration
Step 6: Run Project

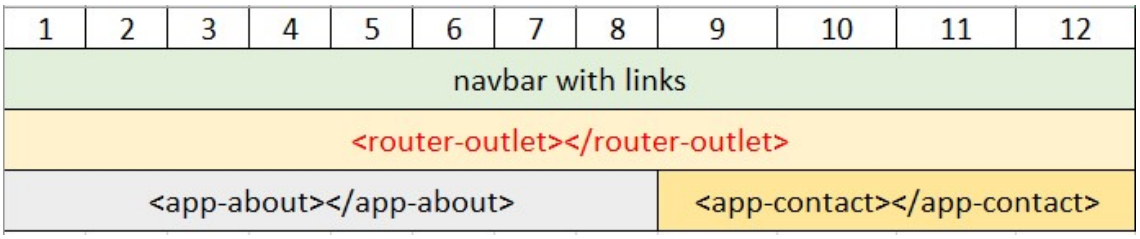
Website Template



Step 3: Create Custom Components
ng g c [component name]

Use the following commands to create required Angular Components:
ng g c home → check home.component.ts for selector: app-home
ng g c about → check about.component.ts for selector: app-about
ng g c services → check services.component.ts for selector: app-services
ng g c contact → check contact.component.ts for selector: app-contact
ng g c gallery → check gallery.component.ts for selector: app-gallery

Step 4: Update app.component.html

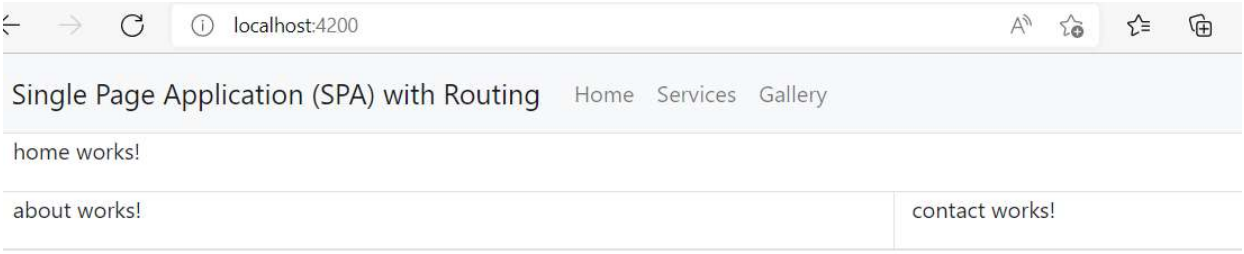



```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container">
    <a class="navbar-brand" href="#">Single Page Application (SPA) with Routing</a>
    <div class="collapse navbar-collapse" id="navbarText">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item">
          <a class="nav-link" routerLink="/home">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" routerLink="/services">Services</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" routerLink="/gallery">Gallery</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
<div class="container border">
  <div class="row border-bottom">
    <div class="col-md-12">
      <router-outlet></router-outlet>
    </div>
  </div>
  <div class="row border-bottom">
    <div class="col-md-8">
      <app-about></app-about>
    </div>
    <div class="col-md-4 border-start">
      <app-contact></app-contact>
    </div>
  </div>
</div>
```

Step 5: Update app-routing.module.ts file

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

import { HomeComponent } from './home/home.component';
import { ServicesComponent } from './services/services.component';
import { GalleryComponent } from './gallery/gallery.component';

const routes: Routes = [

  { path: '', component: HomeComponent },
  { path: 'home', component: HomeComponent },
  { path: 'services', component: ServicesComponent },
  { path: 'gallery', component: GalleryComponent },

];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule {
```

Step 6: Run the Project



Click on Home, Services and Gallery links and observe dynamic display of content below navbar.

Week 6: Data Binding (Part 1)

(String Interpolation, Event Binding and Property Binding)

Objective:

To create Angular Project and Implement Data Binding Concepts

Procedure:

Step 1: Create New Angular Project with Routing Option
Step 2: Install bootstrap and configure to the Project
Step 3: Update app.component.html file with navbar and add Student Details using Data Binding Concept.
Step 4: Declare variables in app.component.ts file
Step 5: Run Project

Web Page – Template

Step 3: Update app.component.html

```
<nav class="navbar navbar-expand-lg navbar-light bg-primary">
  <div class="container-fluid">
    <a class="navbar-brand" [href]="companyUrl" target="new"><img
[src]="logoUrl"></a>
  </div>
</nav>
<br>
<div class="container ">
  <div class="row ">
    <div class="col-md-12">
      <table>
        <tr>
          <td colspan="2">
            <h5>Student Details</h5>
          </td>
        </tr>
        <tr>
          <td style="width:100px;">Roll No.: </td>
          <td class="text-primary">{{student.rollNo}}</td>
        </tr>
        <tr>
          <td style="width:100px;">Name: </td>
          <td class="text-primary">{{student.name}}</td>
        </tr>
        <tr>
          <td colspan="2">
            <button (click)="getEmailId()">Get Email Id</button>
          </td>
        </tr>
      </table>
    </div>
  </div>
</div>
```

Step 3: Update app.component.ts

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

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

  // Property Binding Implementation
  logoUrl:string = "/assets/MRU-Logo.webp";
  companyUrl:string = "https://www.mallareddyuniversity.ac.in";

  // String Interpolation Implementation
  student = {
    rollNo: "2011CS010223",
    name: "PALADUGU NIKETH",
    emailId: "2011cs010223@mallareddyuniversity.ac.in"
  }

  //Event Binding Implementation
  getEmailId() {
    alert(this.student.emailId);
  }

}
```

Step 5: Run the Project

Note:
Use Your Personal Details

Week 7: Data Binding (Part 2)

(Class Binding, Style Binding and Two-way Binding)

Objective:

To create Angular Project and Implement Data Binding Concepts

Procedure:

Step 1: Create New Angular Project with Routing Option
Step 2: Install bootstrap and configure to the Project
Step 3: Update app.component.html file with navbar and Data Binding Examples
Step 4: Declare variables in app.component.ts file
Step 5: Add import {FormsModule} from '@angular/forms'; in app.modules.ts (Required for ngModel)
Step 6: Run Project

Web Page – Template

Class Binding Example

Style Binding Example

Two-way Binding Example

Enter Name:

Step 3: Update app.component.html

```
<nav class="navbar navbar-expand-lg navbar-light bg-primary">
  <div class="container-fluid">
    <a class="navbar-brand" [href]="companyUrl" target="new"><img [src]="logoUrl"></a>
  </div>
</nav>
<br>
<div class="container">
    <div class="row">
        <div class="col-md">
            <h4 [class]="className">Class Binding Example</h4>
        </div>
    </div>
    <div class="mt-4"></div>
    <div class="row">
        <div class="col-md">
            <h4 [style.color]="styleName">Style Binding Example</h4>
        </div>
    </div>
    <div class="mt-4"></div>
    <div class="row">
        <div class="col-md">
            <h4 >Two-way Binding Example</h4>
            Enter Name:
            <input type="text" [(ngModel)]="name">
            <br>
            {{name}}
        </div>
    </div>
</div>
```

Step 4: Update app.component.ts

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

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

  logoUrl:string = "/assets/MRU-Logo.webp";
  companyUrl:string = "https://www.mallareddyuniversity.ac.in";
  className:string = 'text-primary';
  styleName:string = 'red';
  name:string="";

  constructor() { }

  ngOnInit(): void {
  }

}
```

Step 5: Run the Project

Week 8: Directives

(*ngIf, *ngFor, ngSwitch, ngStyle and ngClass)

Objective:

To create Angular Project and Implement Angular Directives

Procedure:

Step 1: Create New Angular Project with Routing Option
Step 2: Install bootstrap and configure to the Project
Step 3: Update app.component.html file Directives Examples
Step 4: Declare variables in app.component.ts file
Step 5: Run Project

Step 3: Update app.component.html

```
<!-- Implementation of *ngIf, *ngFor -->
<div class="container">
    <div class="row">
        <h1 *ngIf="showHeader==true">Directives</h1>

        <h2>*ngIf Directive</h2>
        <div *ngIf = "contacts != null" >some content</div>

        <hr>
        <div *ngIf="contacts.length>0">
            <h2>*ngFor Directive</h2>
            <table class="table">
                <thead>
                    <th>Id No</th>
                    <th>Name</th>
                    <th>Email</th>
                </thead>
                <tbody>
                    <tr *ngFor="let contact of contacts">
                        <td> {{contact.idno}} </td>
                        <td> {{contact.name}} </td>
                        <td> {{contact.email}} </td>
                    </tr>
                </tbody>
            </table>
        </div>
```

<!-- Implementation of ngSwitch -->

```
<hr>
<div class="row">
    <div class="col-md">
        <h2>ngSwitch Directive</h2>
        <div [ngSwitch]="contacts.length">
            <span *ngSwitchCase="1"> One Contact is there in the
                Contacts Array </span>
            <span *ngSwitchCase="2"> Two Contact is there in the
                Contacts Array </span>
            <span *ngSwitchDefault> More than Two Contacts are there in
                Contacts Arrays </span>
        </div>
    </div>
</div>
```

<!-- Implementation of ngClass, ngStyle -->

```
        <hr>
        <h2>ngStyle Directive</h2>
        <p> This directive updates the styles of an HTML element.</p>
        <div class="col-md" [ngStyle]="{'background-color':'green'}">Hello</div>
        <br>
        <div class="mt-4"></div>
        <div class="col-md" [ngStyle]="{'background-color':country === 'UK' ? 'yellow' : 'red' }">
            {{country}}
        </div>

        <hr>
        <h2>ngClass Directive</h2>
        <p>
            The NgClass directive allows you to set the CSS class dynamically for a DOM element.
            Implementing same example with ngClass is easy and there is no need of any method.
        </p>
        <div class="col-md" [ngClass]="{'text-danger': country === 'UK'}"> {{country}}</div>
    </div>

    <div class="mt-4"></div>

</div>
```

Step 4: Update app.component.ts

src > app > TS app.component.ts > ...

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'CustomDirective';

  showHeader:boolean = true;
  data:any;
  country:any= 'UK';

  contacts:any = [
    {idno:"101",name: "Sravani", email:"sravani@gmail.com"},
    {idno:"102",name: "Abhinandan", email:"abhinandan@gmail.com"},
    {idno:"103",name: "Maruti", email:"maruti@gmail.com"},
    {idno:"104",name: "Kamalesh", email:"kamalesh@gmail.com"}
    ];

}
```

Step 5: Run the Project

Output:

Directives

*ngIf Directive

some content

*ngFor Directive

Id No	Name	Email
101	Sravani	sravani@gmail.com
102	Abhinandan	abhinandan@gmail.com
103	Maruti	maruti@gmail.com
104	Kamalesh	kamalesh@gmail.com

ngSwitch Directive

More than Two Contacts are there in Contacts Arrays

ngStyle Directive

This directive updates the styles of an HTML element.

Hello

UK

ngClass Directive

The NgClass directive allows you to set the CSS class dynamically for a DOM element. Implementing same example with ngClass is easy and there is no need of any method.

UK

Week 9: Browser Events

(Click, KeyUp, Change Events)

Objective:

To collect Form data using Brower Events

Procedure:

Step 1: Create New Angular Project
Step 2: Install bootstrap and configure to the Project
Step 3: Create HTML Form in app.component.html file
Step 4: Collect and Validate Data using Brower Events in app.component.ts file
Step 5: Run Project

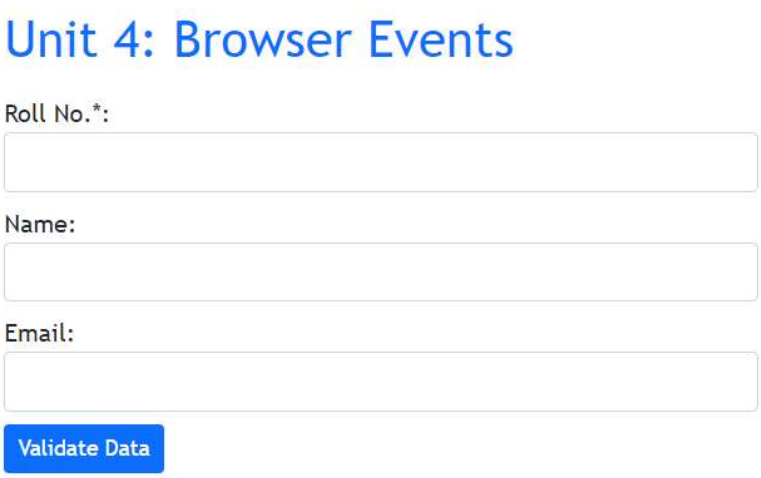
Step 3: Create HTML Form

```
<div class="row">
    <div class="col-md-4">
        <div class="mt-2" >
            <i class="text-danger">{{eventLog}}</i>
        </div>
        <div class="mt-2" >
          Roll No.*: <input type="text"
          [(ngModel)] = "RollNo"
          class="form-control" />
      </div>
        <div class="mt-2" >
            Name: <input type="text"
            (keyup)="nameKeyUpEventHandler($event)"
            class="form-control" />
        </div>
        <div class="mt-2" >
            Email: <input type="text"
            (change)="emailChangeEventHandler($event)"
            class="form-control" />
        </div>
        <div class="mt-2" >
          <Button  class="btn btn-sm btn-primary"
          (click)="validateData()"
          > Validate Data</Button>
      </div>
 </div>
```

Step 4: Collect and Validate Data

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

@Component({
  selector: 'app-browser-events',
  templateUrl: './browser-events.component.html',
  styleUrls: ['./browser-events.component.css']
})
export class BrowserEventsComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

  eventLog:any="";

  RollNo:any="";
  Name:any="";
  EmailId:any="";


nameKeyUpEventHandler(evt:any){
  this.Name = evt.target.value;
  this.eventLog = "name KEYUP event fired. New Name is '" + this.Name + "'";
}

emailChangeEventHandler(evt:any){
  this.EmailId = evt.target.value;
  this.eventLog = "email CHANGE event fired. New Email is '" + this.EmailId + "'";
}

validateData(){
  if (this.RollNo !=""){
    this.eventLog = "Roll No.: " +  this.RollNo;
  }
  else{
    this.eventLog = "Enter Roll No.";
  }

}
```

Output:



Week 10: Built-in Pipes

Objective:

To implement Pipe concept in Expressions to modify the result of expression for display in a view.

Procedure:

Step 1: Create New Angular Project
Step 2: Install bootstrap and configure to the Project
Step 3: Create HTML and Bind Data along with Pipes in app.component.html
Step 4: Declare Variables in app.component.ts file
Step 5: Run Project

Step 3: Create HTML and Bind Data along with Pipes in app.component.html

```
<div>
    Uppercase: {{"malla reddy university" | uppercase }} <br>
    Lowercase: {{"HELLO WORLD" | lowercase}} <br>
    Date: {{today | date:'mediumDate'}} <br>
    Date: {{today | date:'shortTime'}} <br>
    Number: {{3.1415927 | number:'2.1-5'}} <br>
    Number: {{28.25 | number:'3.1-3'}} <br>
    Currency: {{125.25 | currency:'INR'}} <br>
    Currency: {{2158.925 | currency}} <br>
    Json: {{jsonObject | json}} <br>
    PercentPipe: {{0.8888 | percent: '2.2'}} <br>
    SlicePipe: {{"hello world" | slice:0:5}} <br>
    SlicePipe: {{days | slice:1:4}} <br>
</div>
```

Step 4: Declare Variables in app.component.ts file

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

@Component({
  selector: 'app-expression-pipes',
  templateUrl: './expression-pipes.component.html',
  styleUrls: ['./expression-pipes.component.css']
})
export class ExpressionPipesComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

  today = Date.now();
  jsonObject = [{ title: "mytitle" }, { title: "Programmer" }];
  days = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']

}
```

Output:

Uppercase: MALLA REDDY UNIVERSITY
Lowercase: hello world
Date: Jun 17, 2022
Date: 9:38 AM
Number: 03.14159
Number: 028.25
Currency: ₹125.25
Currency: $2,158.93
Json: [{ "title": "mytitle" }, { "title": "Programmer" }]
PercentPipe: 88.88%
SlicePipe: hello
SlicePipe: Monday,Tuesday,Wednesday

Week 11: Template-Driven Forms

Objective:

To Collect Data through Login Form using Template-Driven Forms

Procedure:

Step 1: Create New Angular Project
Step 2: Install bootstrap and configure to the Project
Step 3: Create Login Form in app.component.html
Step 4: Collect Data on Form Submission in app.component.ts file
Step 5: Run Project

Step 3: Create Login Form in **app.component.html**

```
<form ngNativeValidate
      #userForm="ngForm" (ngSubmit)="userForm.form.valid && onSubmit(userForm)">

    <div class="form-group">
        <label>Email</label>
        <input type="email" class="form-control" name="email"
            [(ngModel)]="email" required >
    </div>
    <div class="form-group">
        <label>Password </label>
        <input type="password" class="form-control" name="password"
            [(ngModel)]="password" minlength="4" maxlength="8"
            pattern="^[0-9]+$" required >
    </div>
    <div class="form-group mt-2">
            <button class="btn btn-sm btn-danger" >Submit</button>
    </div>
</form>
```

Step 4: Collect Data on Form Submission in **app.component.ts** file

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

@Component({
  selector: 'app-template-driven-forms',
  templateUrl: './template-driven-forms.component.html',
  styleUrls: ['./template-driven-forms.component.css']
})
export class TemplateDrivenFormsComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

  email: any;
  password: any;

  formValue:any;
  onSubmit(form: any) {
    this.formValue = form.value;
    console.log(form.value);
  }

}
```

Output:

Week 12: Angular Services

Objective:

To load data from Local JSON file using HttpClient Module

Procedure:

Step 1: Create New Angular Project
Step 2: Install bootstrap and configure to the Project
Step 3: Create Service Component

ng g service [service name]

Step 4: Import HttpClientModule in app.module.ts

import { HttpClientModule } from '@angular/common/http';

Step 5: Create Service in http-service.ts file (Service Component)
Step 6: Consume GetData method of Service Class in app.component.ts file
Step 7: Display Data using *ngFor and Interpolation in app.component.ts file
Step 8: Run Project

Step 5: Create Service in http-service.ts file (Service Component)

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

import { Observable } from 'rxjs';
import { HttpClient } from '@angular/common/http';

@Injectable({
  providedIn: 'root'
})
export class HttpLocalService {

  constructor(private http: HttpClient) { }

  apiURL = "./assets/contacts.json";

  GetContacts(): Observable<any> {
    return this.http.get<any>(this.apiURL)
  }

}
```

Step 6: Consume GetData method of Service Class in app.component.ts file

```
import { Component, OnInit } from '@angular/core';
import { HttpLocalService } from '../http-local.service';

import { HttpClient } from '@angular/common/http';

@Component({
  selector: 'app-test-http-local',
  templateUrl: './test-http-local.component.html',
  styleUrls: ['./test-http-local.component.css']
})
export class TestHttpLocalComponent implements OnInit {
  constructor(
    private httpLocalService:HttpLocalService,
    private httpClient:HttpClient
  ) { }

  ngOnInit(): void {
    this.GetContacts();
  }
   contacts:any = [
     {name: 'Mohan'},
     {name: 'Murali'}
   ]

    //Get JSON Data using Service
   public GetContacts() {
     return this.httpLocalService.GetContacts().subscribe((response: {}) => {
       let data:any = response;
       this.contacts = data;
     });
   }

 }
```

Step 7: Display Data using *ngFor and Interpolation in app.component.ts file

```
<div class="col-md-4">
  <h6>Contacts: </h6>
  <ul>
    <li *ngFor="let contact of contacts">{{contact.Name}}</li>
  </ul>
</div>
```

Output:

Week 13: Working with MongoDB

Objective:
To work on MongoDB including Installation, Configuration, Creation of Database, Managing Collections and Documents.

Procedure:

Step 1: Download MongoDB Community Server (version 5.0.9) from https://www.mongodb.com/try/download/community

Step 2: Install MongoDB Complete Setup along with MongoDB Compass MongoDB Compass is a powerful GUI for querying, aggregating, and analysing your MongoDB data in a visual environment.
(https://downloads.mongodb.com/compass/mongodb-compass-1.32.4-win32-x64.exe)

Step 3: Set MongoDB in the windows path environment

Step 4: Working with MongoDB using MongoShell

a) Create **MRU** Database
b) Create Student Collection
c) Insert Student Data (Roll No., and Name)
    i. Single Record
    ii. Two or More Records
d) Show Collection List
e) Show Database List
f) Find First Document
g) Find Document by Criteria
h) Update Student Document by Criteria
i) Remove a Document by Criteria
j) Remove All Documents
k) Drop Student Collection
l) Drop MRU Database

Step 3: Set MongoDB in the windows path environment

- After successful installation, Right-click on '**This PC**' or 'My Computer' and Choose properties
- Choose the `**advance system setting**` options
- Click on **Environment Variables** under Advance section.
- Choose **Path value** under system variables and click Edit button
- Now get your mongo path to your system, where your MongoDB is installed. For example, if you installed MongoDB in C drive, then it your path will be like this: `C:\ProgramFiles\MongoDB\Server\VERSION\bin`
- Copy this path and enter as a new environment value on Edit environment variables page
- Now click on OK and close all active dialog box. Your environment is set, restart your terminal and now enter **mongo**, it will open mongo-shell.

Step 4: Working with MongoDB using MongoShell

- Displaying a List of Databases

  show dbs

- Changing the Current Database

  use MRU

- Check current Database

  db

- Creating Databases

  Your created database (MRU) is not present in list. To display database, you need create collection and need to insert at least one document into it.

  db.createCollection('student')

- Show collections – to see existing Collections

  show collections

- Drop collections

  db.student.drop()

- Deleting Databases

  db.dropDatabase()

- Copying Databases

  db.copyDatabase(‘mru’, ‘mru1’)

- Adding Documents to a Collection
  - To add single document

    db.student.insertOne({'name' : 'raju'})

  - To add Many documents

    db.student.insertMany([{‘name’ : ‘raju’}, {‘name’ : ’rao’}])

- Finding Documents in a Collection
  - To view all documents

    db.student.find({})

  - To view first document

    db.student.findOne({})

  - To view first documents matching Criteria

    db.student.find({'name':'rao’})

- Deleting Documents in a Collection
  - To remove a Single Document

    db.student.remove({'name':'rao’})

  - To remove All Document

    db.student.remove({})

- Updating Documents in a Collection
  - To update first matched document

    db.student.updateOne({'name':'raju'},{$set:{'name':'ram'}})

  - To update first matched document

    db.student.updateMany({'name':'raju'},{$set:{'name':'ram'}})

  - Quit the mongo shell

    exit