FSD LAB

1)Create a simple multiplier Angular JS application which will multiply two numbers and display the result.

```
<html>
<head>
<title>First AngularJS Application</title>
<script src= "~/Scripts/angular.js"></script>
</head>
<body ng-app >
<h1>First AngularJS Application</h1>
Enter Numbers to Multiply:
<input type="text" ng-model="Num1" /> x <input type="text" ng-model="Num2" />
<span ng-model="Num1 * Num2">{{Num1 * Num2}}</span>
</body>
</html>
```

2)Write an angular program to share data from parent to child component via @Input Decorator

Create 2 components parent.component.ts and child.component.ts

parent.component.ts

```
import { Component } from '@angular/core';
@Component({
selector: 'app-parent',
template: `
<app-child [childMessage]="parentMessage"></app-child>
styleUrls: ['./parent.component.css']
})
export class ParentComponent{

parentMessage = "message from parent";
constructor() { } }
```

<u>Child.component.ts</u>

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

@Component({
selector: 'app-child',
template: ` Say {{ message } `,
styleUrls: ['./child.component.css']
})

export class ChildComponent {

@Input() childMessage: string;

constructor() { }
}
```

3)1. Write an angular program to share data from child to parent component via @Output Decorator.

in parent.component.ts file

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

@Component({
selector: 'app-parent',
template: `
Message: {{message}}
<app-child (messageEvent)="receiveMessage($event)"></app-child>

`,
styleUrls: ['./parent.component.css']
})
```

```
export class ParentComponent {
constructor() { }

Message:string;
receiveMessage($event) {
this.message = $event
}
}
```

code in child.component.ts file

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

@Component({
selector: 'app-child',
template: `
<button (click)="sendMessage()">Send Message</button>
`,
styleUrls: ['./child.component.css']
})

export class ChildComponent {
message: string = "Hey Angular!"
@Output() messageEvent = new EventEmitter<string>();
constructor() { }
sendMessage() {
this.messageEvent.emit(this.message)
}
}
```

4)Write an angular program to share data from child to parent component via ViewChild with AfterViewInit

in parent.component.ts file

```
import { Component, ViewChild, AfterViewInit } from '@angular/core';
import { ChildComponent } from "../child/child.component";
@Component({
selector: 'app-parent',
template: `
Message: {{ message }}
<app-child></app-child>
`,
styleUrls: ['./parent.component.css']
})
export class ParentComponent implements AfterViewInit {
@ViewChild(ChildComponent) child;
constructor() { }
Message:string;
ngAfterViewInit() {
this.message = this.child.message
}
}
```

in child.component.ts

```
import { Component} from '@angular/core';
@Component({
selector: 'app-child',
template: ` `,
styleUrls: ['./child.component.css']
})
export class ChildComponent {
message = 'Hello Angular!';
constructor() { }
}
```

5) Write an inline template to display college details(name ,department name,and address etc)

1st step:

ng new **myapp**

Open app.component.ts file

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

@Component({
selector: 'app-root',
template:`<h1> My Location Details</h1>
<div>
<div class="name">
Place:{{ user.name }}
</div>
<div class="address">
Name:{{user.address}}
<div class="department">
Department:{{user.department}}
</div>
</div>
</div>`,
styles: [`
h1 { font-weight: normal; color:red;}
.name {font-weight: bold; color:blue; }
.address {font-weight: bold; color:green;}
.department {font-weight: bold; color:grey;}
`]
})
export class AppComponent {
title = 'myapp';
user={name:"Ajay Pandey",department: "AIML",
address:"Hyderbad"};
}
```

6) Write an external template to display student details(id and name.etc)

Step 1: create a new application as:

```
ng new myapp
```

Step 2: Got to /myapp/src/app

Step 3: Open the file app.component.ts

```
import { Component } from '@angular/core';
@Component({
selector: 'app-root',
templateUrl: './app.component.html',
styleUrls: ['./app.component.css']
})
export class AppComponent {
title = 'myapp';
student_detail={id:101, name:"Sanjay Paul"}
}
```

Step 4: Open app.component.css and write

```
h1 { font-weight: normal; color:red;}
.stud_id {font-weight: bold; color:blue;}
.stud_name {font-weight: bold; color:green;}
```

Step 5: Open app.component.html and write:

app.component.html

```
<h1> This is External Template Example</h1>
<div>
<div class="stud_id">
Student ID:{{student_detail.id}}
</div>
<div class="stud_name">
Student Name:{{student_detail.name}}
</div>
</div>
```

Step 6: Type the following command execute

ng serve –open

7) Write an angular program to use built-in pipes: Uppercase, Lowecase, Date, Currency, Json and Slice

app.component.ts file

```
import { Component } from '@angular/core';
@Component({
selector: 'app-root',
templateUrl: './app.component.html',
styleUrls: ['./app.component.css']
})
export class AppComponent {
title = 'My First App';
todaydate = new Date();
jsonval = {name:'Rox', age:'25', address:{a1:'Mumbai', a2:'Karnataka'}};
months = ["Jan", "Feb", "Mar", "April", "May", "Jun",
"July", "Aug", "Sept", "Oct", "Nov", "Dec"];
}
```

App.component.html

```
<!--The content below is only a placeholder and can be replaced.-->
<div style = "width:100%;">
<div style = "width:40%;float:left;border:solid 1px black;">
<h1>Uppercase Pipe</h1>
<b>{{title | uppercase}}</b><br/>
<h1>Lowercase Pipe</h1>
<b>{{title | lowercase}}</b>
<h1>Currency Pipe</h1>
<b>{{6589.23 | currency:"USD"}}</b><br/>
<b>{{6589.23 | currency:"USD":true}}</b> //Boolean true is used to get the sign of the
currency.
<h1>Date pipe</h1>
<b>{{todaydate | date:'d/M/y'}}</b><br/>
<b>{{todaydate | date:'shortTime'}}</b>
<h1>Decimal Pipe</h1>
<b>{{ 454.78787814 | number: '3.4-4' }}</b> // 3 is for main integer, 4 -4 are for integers
to be
displayed.
```

```
</div>
<div style = "width:40%;float:left;border:solid 1px black;">
<h1>Json Pipe</h1>
<b>{{ jsonval | json }}</b>

<h1>Percent Pipe</h1>
<b>{{00.54565 | percent}}</b>
<h1>Slice Pipe</h1>
<b>{{months | slice:2:6}}</b>
// here 2 and 6 refers to the start and the end index
</div>
</div>
```

8)1. Create custom PIPES to multiply two numbers.

Step 1 − First, create a file called **multiplier.pipe.ts**.

• Step 2 − Place the following code in the above created file.

```
import { Pipe, PipeTransform } from '@angular/core';
@Pipe({
name: 'multiplier'
})
export class MultiplierPipe implements PipeTransform {
transform(value: number, multiply: string): number {
let mul = parseFloat(multiply);
return mul * value
}}
```

App.component.ts

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

in the app.module.ts file.

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { MultiplierPipe } from './multiplier.pipe';
@NgModule({
declarations: [
AppComponent,
MultiplierPipe
],
imports: [
BrowserModule,
AppRoutingModule
],
providers: [],
bootstrap: [AppComponent]
})
export class AppModule { }
```

app.component.html file.

```
<h1>Nothing is working <p>Multiplier: {{2 | multiplier: "10"}}</p> </h1>
```

9)Write an angular example to demonstrates ng-if, ng-readonly, and ng-disabled directives.

Step 1: At command prompt type:

```
ng new myapp
```

Step 2: Open to src/app.component.ts

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

```
title = 'myapp';
}
```
Step 3: Open to src/app.component.html
```
<!DOCTYPE html>
<html>

<head>
<script src="~/Scripts/angular.js"></script>
<style>
div {
width: 100%;
height: 50px;
display: block;
margin: 15px 0 0 10px;
}
</style>
</head>
<body ng-app ng-init="checked=true" >
Click Me: <input type="checkbox" ng-model="checked" /> <br />
<div>
New: <input ng-if="checked" type="text" />
</div>
<div>
Read-only: <input ng-readonly="checked" type="text" value="This is read-only." />
</div>
<div>
Disabled: <input ng-disabled="checked" type="text" value="This is disabled." />
</div>
</body>
</html>
```
Step 4: Open cmd type ng serve --open

10)Write an angular program to perform Arithmetical operations using ngSwitch

1.Create a new Angular component using the Angular CLI command:

```
ng generate component arithmetic
```

2.Update the arithmetic.component.html template file with the following content:

```
<div>
<h1>Arithmetic Operations</h1>
<input [(ngModel)]="num1" placeholder="Enter Number 1" type="number">
<input [(ngModel)]="num2" placeholder="Enter Number 2" type="number">
<br><br>
<button (click)="add()">Addition</button>
<button (click)="subtract()">Subtraction</button>
<button (click)="multiply()">Multiplication</button>
<button (click)="divide()">Division</button>
<br><br>
<p>Result: {{ result }}</p>
</div>
```

3.Update the arithmetic.component.ts TypeScript file with the following content:

```
import { Component } from '@angular/core';
@Component({
selector: 'app-arithmetic',
templateUrl: './arithmetic.component.html',
styleUrls: ['./arithmetic.component.css']
})
export class ArithmeticComponent {
num1: number;
num2: number;
result: number;
add() {
this.result = this.num1 + this.num2;
}
subtract() {
this.result = this.num1 - this.num2;
}
multiply() {
```

```
this.result = this.num1 * this.num2;
}
divide() {
if (this.num2 === 0) {
this.result = NaN;
} else {
this.result = this.num1 / this.num2;
}
}
}
```

4.Update the arithmetic.component.css CSS file with the following content (optional):

```
input {
margin: 5px;
}
button {
margin: 5px;
}
```