Angular 8 –Sep 23 2019–Teachers Notes. (Level 1- LAB BOOK) BY M.H. SHOIAB

(The material I was looking forward when I wanted to learn Angular)

Level 1 – For Beginners and Intermediate

Learn Angular 8 in 23 UseCases.

1. **Who developed Angular ?**
    a. **Google**

Note: AngularJS was originally developed in 2009 by Miško Hevery[15] at Brat Tech LLC

2. **What is the Objective of creating angular 2 ?**
    a. **Angular is a JavaScript framework for building UI Components, web applications and apps in JavaScript, html, and TypeScript.**
3. **What you should know before learning Angular?**
    a. **Basic knowledge or know how of Html5, OOJavascript, NodeJS, ES6, TypeScript.**
4. **Salient features of Angular? (We will see the features in detail later in the notes).**
5. **Limitations of Angular ?**
6. **Online editors for Angular - https://stackblitz.com/edit/angular-hqul2y?file=src%2Fapp%2Fapp.component.html**
7. **What is the difference between Angular 1 and Angular 2 to 7 ?**
8. **Angular Setup**
    a. **To install Angular, we require the following –**
    b. **Nodejs -https://nodejs.org/en/download/ - node 10.16.3-64.msi**
    c. **C:\>node –v – To cross check Npm**
    d. **C:\>npm –v – To cross Check Angular CLI**
    e. **C:\> npm install –g @angular/cli**
    f. **C:\> ng version – To cross check**
    g. To install the **Bootstrap 4 CSS Framework** using the following command. You can skip the step of installing bootstrap because we do not need too much of styling here.

        >npm install bootstrap --save

    h. **IDE for writing your code**
    i. **https://code.visualstudio.com/**

    Tips:

        For proxy settings

npm config set proxy http://proxy.company.com:8080
npm config set https-proxy http://proxy.company.com:8080
To remove proxy
Npm config rm proxy
Npm config rm https-proxy
Npm config set registry http://registry.npmjs.org
Typescript
https://www.typescriptlang.org/play/
Issue the following command on dos prompt

c:\>npm install -g typescript

http://brackets.io/ - download brackets UI

UseCase 1 – HelloWorld with Angular

Our First Project

1. C:\>ng new <<project name>>

Now open the Angular app and see how the folder structure looks like in visual studio.

The file structure has the app component and it consists of the following files –

app.component.css

app.component.html

app.component.spec.ts

app.component.ts

app.module.ts

main.ts file.

2. c:\project>ng serve

http://localhost:4200

c:\project>ng serve --host 0.0.0.0 --port 2020 or just ng serve

Tips:

Main.ts – this is the file which loads the app.module.ts file using bootstrap module engine.

App.module.ts – This is our module configuration file, it has two major sections

1. **Declaration – This has entry of our components which we have created.**
2. **Imports – This has entry of all our dependent modules used in our module.**

(Remember before we make an entry of them in both, the said component or module should be declared in the import section of the file on top)

UseCase 2 – Data Handling

Tip: File -> Preferences -> Colour Theme – Light (To change the background colour of the visual studio editor).

App.component.ts

```
import { Component } from '@angular/core';
@Component({
  selector: 'myfirsttag',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
/*
import {
    Component
} from '@angular/core';
@Component({
    selector: "myfirsttag",
    template: '<h2>M.H.Shoiab - 9840135749, FOS-Consultant</h2>'
})
*/
export class AppComponent  {
  constructor() { }

  isavailable = false;

   months =
  ["January", "Feburary", "March", "April", "May", "June",
  "July", "August", "September",
  "October", "November", "December"];

  people: any[] = [
    {
      "name": "Douglas  Pace",
      "age": 35,
      "country": 'MARS'
    },
    {
      "name": "Mcleod  Mueller",
      "age": 32,
      "country": 'USA'
```

```
    },
    {
      "name": "Day  Meyers",
      "age": 21,
      "country": 'HK'
    },
    {
      "name": "Aguirre  Ellis",
      "age": 34,
      "country": 'UK'
    }  ];

  }
```

App.module.ts

```
import { BrowserModule } from
'@angular/platform-browser';
import { NgModule } from
'@angular/core';

import { AppRoutingModule }
from './app-routing.module';
import { AppComponent } from
'./app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Firstproj</title>
  <base href="/">

</head>
<body>
  <myfirsttag></myfirsttag>
</body>
</html>
```

App.component.html

```
<h1>My First Component</h1>
{{months[0]}}
<div *ngFor="let p of people">
{{p?.name?.toUpperCase()}}<br/>
  {{p?.age}}<br/>
  {{p?.country}}<br/>
</div>
<select>
  <option *ngFor="let month of
months">{{month}}</option>
</select>
<div>
  <div>
  <span *ngIf="isavailable;then con1
else con2"></span>
  <ng-template #con1>This is true
part......</ng-template>
  <ng-template #con2>This is false
part......</ng-template>
</div>
<ul *ngFor="let person of people"
    [ngSwitch]="person.country">

  <li *ngSwitchCase="'UK'"
      class="onstate">This is UK-{{
person.name }} ({{ person.country }})
  </li>
 <li *ngSwitchDefault
      class="hsstate">{{ person.name }}
({{ person.country }})
  </li>
</ul>
<input type="text" [value]="months[0]">
<img [src]="imgsrc" alt="image not
found">
<button
```

		[disabled]="!isAvailable">{{months[0]}}</button>

1.The ? in the above app..html page is a special character used to prevent any problem if in case the property name is not present. The ? will ignore and process the rest of the page if the property name is not found.

2. The <ng-template is used inorder to provide a place holder for conditional statements.

UseCase 3 – Event - ngStyle

Step 1: Create a Model file.
Inside the src >> app folder, create one file called Character.ts file and add the following code inside it.

```
// Character.ts

export default class Character {
  actor_name: String;
  character_name: String;
  gender: String;
  status: String;
}

// app.component.ts

import { Component } from '@angular/core';
import Character from './Character';
@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent {
 characters: Character[] = [
  {
     actor_name: 'Peter Dinklage',
     character_name: 'Tyrion Lannister',
     gender: 'Male',
     status: 'Alive'
  },
  {
     actor_name: 'Sean Bean',
     character_name: 'Ned Stark',
     gender: 'Male',
     status: 'Dead'
  }];
```

Step 2: Create HTML view to display data in Angular
Now, the last step is to write the HTML code to display the data.

We will first display the data in Table Format.

Write the following code inside the app.component.html file.

```
<div class="container">
  <h4>NgStyle</h4>
  <div *ngFor="let celeb of characters">
     <p [ngStyle]="{'background-color':celeb.status === 'Dead' ? 'red' : 'green' }">
    {{ celeb.actor_name }} ({{ celeb.character_name }}) is {{celeb.gender}} character and
{{celeb.status}}
     </p>
   </div>
</div>
```

-ngFor

The following exported values can be aliased to local variables.

1. `$implicit: T`: The value of all the individual items in the iterable (`ngForOf`).
2. `ngForOf: NgIterable<T>`: The value of an iterable expression. Useful when the expression is more complicated then property access, for example when using the async pipe (`userStreams | async`).
3. `index: number`: An index of the current item in the iterable.
4. `first: boolean`: True when an item is the first item in the iterable.
5. `last: boolean`: True when an item is the last item in the iterable.
6. `even: boolean`: True when an item has an even index in the iterable.
7. `odd: boolean`: True when an item has an odd index in the iterable.

Now, we use **even** and **odd** local variables to differentiate the rows of the table.

```
<style>
  .odd {
    background-color: yellow;
  }
  .even {
    background-color: cyan;
  }
   .first {
    background-color: red;
  }
  .last {
    background-color: green;
  }

  </style>
    <div class="container">
    <table class="table table-responsive">
      <thead>
          <tr>
            <th>Index</th>
            <th>Actor Name</th>
            <th>Character Name</th>
            <th>Gender</th>
```

```
            <th>Status</th>
          </tr>
      </thead>
      <tbody>
        <tr *ngFor="let character of characters;
                    let i = index;
                    let odd=odd;
                    let even=even;
                    let first = first;
                    let last = last"
                    [ngClass]="{ odd:odd, even:even, first: first, last: last }">
          <td>{{ i }}</td>
          <td>{{ character.actor_name }}</td>
          <td>{{ character.character_name }}</td>
          <td>{{ character.gender }}</td>
          <td>{{ character.status }}</td>
        </tr>
      </tbody>
    </table>
  </div>
```

UseCase 4 – Pipes

Pipes (previously Filters in AngularJS) allow us to essentially create a pure function, which accepts an input and returns a different output via some form of transformation. Angular has many Pipes built-in.

App.component.ts

```
import { Component } from '@angular/core';
@Component({
  selector: 'myfirsttag',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent{
  title = 'firstproj';
  todaydate = new Date();
  jsonval = {name:'Shoiab', age:'43', address:{a1:'chennai', a2:'TamilNadu'}};
}
```

App.component.html

```
<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>
```

UseCase 5 – Component Creation

First issue the below command from the component folder.

C:\SHOIAB\ANGULAR\PROJECTS\firstproj>ng g component new-cmp

```
installing component
   create src\app\new-cmp\new-cmp.component.css
   create src\app\new-cmp\new-cmp.component.html
   create src\app\new-cmp\new-cmp.component.spec.ts
   create src\app\new-cmp\new-cmp.component.ts
   update src\app\app.module.ts
```

In app.module.ts the following entry will be automatically made.

……

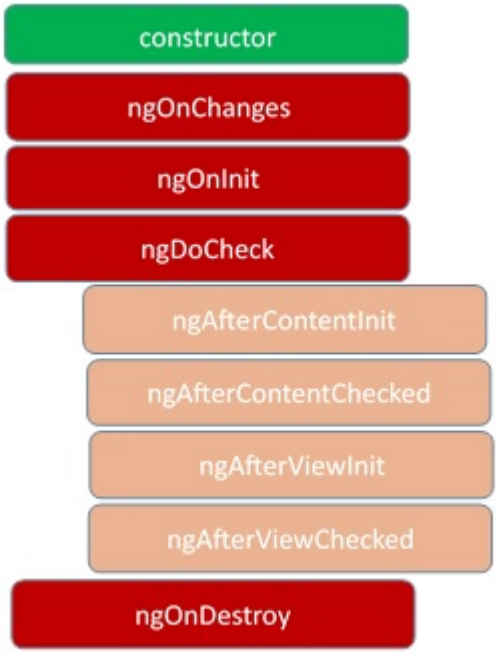
```
import { NewCmpComponent } from './new-cmp/new-cmp.component';
// includes the new-cmp component we created
@NgModule({
   declarations: [
      AppComponent,
      NewCmpComponent // here it is added in declarations and will behave as a child component
   ],
....
```

App.component.html

```
<div style="text-align:center">
        <h1>
           Welcome to {{title}}.
        </h1>
     </div>

     <app-new-cmp></app-new-cmp>
```

UseCase 6 – LiceCycle Hooks



Angular Life Cycle

Every component has a life cycle, which is managed by Angular.

Angular creates it, renders it, creates and renders its children, checks it when its data bound properties change, and destroy it before removing it from the DOM.

Angular offers lifecycle hooks that provide visibility into these key life moments and the ability to act when they occur.

Life Cycle Hooks.
Some Major Events.
ngOnit
ngOnDestroy
ngOnChanges
ngDoCheck
ngAfterContentInit
ngAfterContentChecked
ngAfterViewInit

ngAfterViewChecked
ngOnDestroy

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

@Component({
  selector: 'my-app',
  templateUrl: './app.component.html',
  styleUrls: [ './app.component.css' ]
})
export class AppComponent  implements OnInit,OnDestroy,AfterContentInit,AfterViewInit{
  name = 'Angular';
  constructor(){
    console.log("constructor called.....");
  }
  ngOnInit(){
    console.log("ng on init called....");
  }
  ngOnDestroy(){
    console.log("ng on destroy called.....");
  }
  ngAfterViewInit(){
    console.log("after view initialization");
  }
  ngAfterContentInit(){
    console.log("after content init called......");
  }

}
```

UseCase 7 – IN-OUT Project

Create a Component

Child component html

```
<button (click)="click()" data="hello world...">Click me</button>
```

Child component ts

```
  @Input() data:string;
  @Output() myemit = new EventEmitter();
    counter:number = 0;
    click() { // You can give any function name
        console.log("data....:"+this.data);
```

```
        this.counter = this.counter + 1;
        this.myemit.emit(this.counter);
    }
```

Parent component html file

```
<app-masterchild data="hello world" (myemit)="mainMethod($event)"></app-masterchild>
```

Parent component TS file

```
mainMethod(counter){
  console.log("counter..:"+counter);
}
```

Use Case 8 – LifeCycle – ng Changes

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

@Component({
  selector: 'my-app',
  templateUrl: './app.component.html',
  styleUrls: [ './app.component.css' ]
})
export class AppComponent  {
  private number:number=1010122;

get counter(){
  return this.number;
}
set counter(value){
  this.number=value;
}

increment(){
  this.counter++;
}
decrement(){
  this.counter--;
}
}
```

Mychild.component.ts

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

@Component({
```

```
  selector: 'app-mychild',
  templateUrl: './mychild.component.html',
  styleUrls: ['./mychild.component.css']
})
export class MychildComponent implements OnInit,OnChanges {

  @Input() myNewNumber:number;
  // @Input()
  // set myNewNumber(value:number){
  //   this.myNumber=value;
  // }
  // get myNewNumber(){
  //   return this.myNumber;
  // }
  ngOnInit(){
    console.log('init...',this.myNewNumber);
  }

  ngOnChanges(changes:SimpleChanges){
    console.log("entered into onchange....");
    const newNumberChange:SimpleChange=changes.myNewNumber;
console.log('Previious Value-',newNumberChange.previousValue);
console.log('Next Value-',newNumberChange.currentValue);
  }
}
```

Mychild.component.html

```
<p>
    mychild works!

    {{myNewNumber}}
</p>
```

App.component.html

```
<hello name="{{ name }}"></hello>
<p>
  Start editing to see some magic happen :)
</p>
<div>
<button (click)="increment()">Addition</button>
<app-mychild [myNewNumber]="counter"></app-mychild>
<button (click)="decrement()">Subtraction</button>
</div>
```

UseCase 9 - LifeCycle

Child component

```
import { Component,
OnInit,OnChanges,DoCheck,AfterContentChecked,AfterContentInit,AfterViewInit,AfterViewChecked,OnDestr
oy, SimpleChanges, Input, Output, EventEmitter } from '@angular/core';

@Component({
  selector: 'app-lifecyclehooks',
  templateUrl: './lifecyclehooks.component.html',
  styleUrls: ['./lifecyclehooks.component.css']
})
export class LifecyclehooksComponent  implements OnChanges,OnInit,OnDestroy,AfterViewInit{

  @Input() simpleInput:string;
  constructor() {
    console.log("con called....");
  }
  ngOnInit(){
    console.log("on init called....");
  }
  ngOnDestroy(){
    console.log("on destroy called....");
  }
  ngAfterViewInit(){
    console.log("on after view init called....");
  }
  p:string;c:string;
  ngOnChanges(changes:SimpleChanges){
    console.log("on changes called...");
    for(let propname in changes){
      let change=changes[propname];
      let current=JSON.stringify(change.currentValue);
      let previous=JSON.stringify(change.previousValue);
      this.p=previous;this.c=current;
      console.log("previous..:"+previous+":Current..:"+current);
    }
  }
}
```

Child component html page

```
you entered :{{simpleInput}}
<br/>
Previous:{{p}}
<br/>
Changed:{{c}}
```

Main component ts file

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

@Component({
  selector: 'app-root',
```

```
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  userText:string="hello";
}
```

Main component html page

```
Your Text:<input type="text" [(ngModel)]='userText'/>

<br/>
<app-lifecyclehooks [simpleInput]='userText'></app-lifecyclehooks>
```

Use Case 10 – Event Handling

App.component.ts

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

@Component({
  selector: 'myfirsttag',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit,OnDestroy,OnChanges {
  isAvailable=true;
  months=["jan","feb","march"];

  constructor(){
    console.log("constructor called....");
  }
    values="";values2="";values3="";values4;
 title="hello world.......";
 clickMe(){
    console.log("clicked.....");
    this.title="hello world changed....";
  }
clickMe2(){
    console.log("clicked...from additional method..");
  }

    onKey(event: KeyboardEvent){
      console.log(e.target.value);
     // this.values += (<HTMLInputElement>event.target).value + ' | ';
    }
    onKey2(value:string){
```

```
      this.values2=value;
    }
    onBlur(value:string){
      this.values3=value;
    }
    onEnter(value:string){
      this.values4=value;
    }
addMonths(m: string) {
    if (m) {
      this.months.push(m);
    }
  }

}
```

App.component.html

```
<!--The content below is only a placeholder and can be replaced.-->
<h1>My First Component</h1>

<button (click)="clickMe()">Click Me</button>
{{title}}
<button (click)="clickMe();clickMe2()">Click Me</button>

<input (keyup)="onKey($event)">
<p>OnKey:{{values}}</p>

<input #box
(keyup)="onKey2(box.value)"
(blur)="onBlur(box.value)"
(keyup.enter)="onEnter(box.value)"
>
<p>Keyup:{{values2}}</p>
<p>Blur:{{values3}}</p>
<p>Enter:{{values4}}</p>

<input #m
      (keyup.enter)="addMonths(m.value)"
      (blur)="addMonths(m.value); m.value='' ">

    <button (click)="addMonths(m.value)">Add</button>

    <ul><li *ngFor="let m of months">{{m}}</li></ul>


<router-outlet></router-outlet>
```

UseCase 10 – Editable Table

App.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
```

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

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

Step 2 – download bootstrap and copy the css and js folder to assets folder in src

Step 3 – Edit angular.json

```
"build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist/edittableproj",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "src/tsconfig.app.json",
            "assets": [
              "src/favicon.ico",
              "src/assets"
            ],
            "styles": [
              "src/styles.css",
              "src/assets/css/bootstrap.css"
            ],
```

Step 4

App.component.ts

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

```
  TableData:any=[];
ShowEditTable:boolean=false;
EditRowID:any='';
constructor(){
this.TableData=[
{id:1,name:'ramu',mobile:'989898989',email:'ram@aa.com'},
{id:2,name:'somu',mobile:'989898989',email:'sam@aa.com'},
{id:3,name:'ramesh',mobile:'989898989',email:'ram@aa.com'},
{id:4,name:'somesh',mobile:'989898989',email:'ram@aa.com'}
];
}

Edit(val){
  this.EditRowID=val;
}
}
```

App.component.html

```
<div class="row">
  <div class="col-lg-l3">
    <table class="table table-bordered">
      <thead>
      <th>Name</th>
      <th>Mobile</th>
      <th>Email</th>
      </thead>

    <tr *ngFor="let user of TableData">

        <td *ngIf="user.id===EditRowID">
            <input type="text" [(ngModel)]="user.name">
        </td>
        <td *ngIf="user.id !==EditRowID" (click)="Edit(user.id)">{{user.name}}</td>

      <td>{{user.mobile}}</td>
      <td>{{user.email}}</td>
    </tr>
  </table>
</div>
</div>
<div class="row">
  <pre>
    {{TableData|json}}
  </pre>
</div>
```

UseCase 11 – Forms - Template Driven Forms

With a template driven form, most of the work is done in the template; and with the model driven form, most of the work is done in the component class.

App.component.ts

```
import { Component,OnInit } from '@angular/core';
import { FormsModule } from '@angular/forms';
interface Data{
  uname:string;
  emailid:string;
  password:string;
}

@Component({
  selector: 'myfirsttag',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent implements OnInit{
  constructor(){
  }
  ngOnInit(){
  }
  value="welcome...:"
  onClickSubmit(data:Array<Data>){
    console.log(data["uname"]);
    console.log(data["emailid"]);
    console.log(data["password"]);
}
}
```

App.module.ts

```
import { BrowserModule } from
'@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-
routing.module';
import { AppComponent } from './app.component';
import { FormsModule } from '@angular/forms';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
```

App.component.html

```
<form #userlogin = "ngForm" (ngSubmit) =
"onClickSubmit(userlogin.value)" >
    <input type="text" name="uname"
placeholder="UserName" ngModel>
    <br>
    <input type = "text" name = "emailid"
placeholder = "emailid" ngModel>
    <br/>
    <input type = "password" name = "password"
placeholder = "passwd" ngModel>
    <br/>
    <input type = "submit" value = "submit">
 </form>
```

```
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

*

UseCase 12 – Forms – ReactiveForms or Model Driven Form

In the model driven form, we need to import the ReactiveFormsModule from @angular/forms and use the same in the imports array.

App.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { ReactiveFormsModule } from '@angular/forms';

@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    ReactiveFormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

app.component.html

```
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl } from '@angular/forms';
interface Data{
  name:string;
  email:string;
}
@Component({
  selector: 'app-rform-rformsimple',
  templateUrl: './rform-rformsimple.component.html',
  styleUrls: ['./rform-rformsimple.component.css']
})
export class RformRformsimpleComponent implements OnInit {

  constructor() { }
  employeeForm:FormGroup;
  ngOnInit() {
    this.employeeForm=new FormGroup({
     name:new FormControl(),
     email:new FormControl(),
```

```
    });
  }
  email:string;
  onClickSubmit(data:Array<Data>){
    this.email=data["email"];
    console.log(data["name"]);
    console.log(data["email"]);
}
  onLoadDataClick():void{
    //this.employeeForm.setValue({//for all values updation
    this.employeeForm.patchValue({
      //for some values updation
    name:'ramu',
    email:'aa@gmail.com',
  });
  }
}
```

```
<div>
  <form [formGroup]="employeeForm"
           (ngSubmit) = "onClickSubmit(employeeForm.value)" >
      <input type="text" placeholder="name"
           formControlName="name">
     <input type="text" placeholder="email"
           formControlName="email">
     <br/>

     <input type="submit"
          [disabled] = "!employeeForm.valid"
          value="Log In">
  </form>
  <input type="button"
          (click) = "onLoadDataClick()"
          value="Load Data">
</div>
<p>
  Email entered is : {{email}}
</p>
```

UseCase 13 - SubForms

```
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl,Validators } from '@angular/forms';
interface Skill{
  skillName:string;
  experienceInYears:number;
  proficiency:string;
}
interface Data{
  name:string;
  email:string;
  skills:Array<Skill>;
}
@Component({
```

```
  selector: 'app-rform2',
  templateUrl: './rform2.component.html',
  styleUrls: ['./rform2.component.css']
})
export class Rform2Component implements OnInit {

  constructor() { }
employeeForm:FormGroup;
  ngOnInit() {
    this.employeeForm=new FormGroup({
     name:new FormControl(),
     email:new FormControl(),
     skills:new FormGroup({
       skillName:new FormControl(),
       experienceInYears:new FormControl(),
       proficiency:new FormControl()
     })
    });
  }
  email:string;
  onClickSubmit(data:Array<Data>){
    this.email=data["email"];
    console.log(data["name"]);
    console.log(data["email"]);
    console.log(data["skills"]["skillName"]);
    console.log(data["skills"]["experienceInYears"]);
    console.log(data["skills"]["proficiency"]);
   // console.log(data["password"]);
}
  onLoadDataClick():void{
    //this.employeeForm.setValue({//for all values updation
    this.employeeForm.patchValue({
      //for some values updation
    name:'ramu',
    email:'aa@gmail.com',
    skills:({
      skillName:'c#',
      experienceInYears:5,
      proficiency:'beginner'
    })
  });
  }
}
```

```
<div>
  <form [formGroup]="employeeForm"
            (ngSubmit) = "onClickSubmit(employeeForm.value)" >
      <input type="text" placeholder="name"
           formControlName="name">
     <input type="text" placeholder="email"
           formControlName="email">
     <br/>
     <div formGroupName="skills">
     <input type="text"  placeholder="skillname"
           formControlName="skillName">
```

```
     <br/>
     <input type="text"  placeholder="experienceInYears"
     formControlName="experienceInYears">
    <br/>
    <input type="radio" value="beginner"  name="proficiency"
    formControlName="proficiency">Beginner
    <input type="radio" value="intermediate"  name="proficiency"
    formControlName="proficiency">intermediate
    <input type="radio" value="advanced"  name="proficiency"
    formControlName="proficiency">Advanced
     </div>
     <input type="submit"
          [disabled] = "!employeeForm.valid"
          value="Log In">
  </form>
  <input type="button"
          (click) = "onLoadDataClick()"
          value="Load Data">
</div>
<p>
  Email entered is : {{email}}
</p>
```

UseCase 14 – Form Validation

```
import { Component,OnInit } from '@angular/core';
import { FormGroup, FormControl,Validators } from '@angular/forms';

interface Data{
  uname:string;
  emailid:string;
  password:string;
}

@Component({
  selector: 'myfirsttag',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent implements OnInit{
  constructor(){
  }
  todaydate;
   componentproperty;
   uname;
   emailid;
   formdata;
  ngOnInit(){
     this.formdata = new FormGroup({
         uname:new FormControl("username",Validators.compose([
           Validators.required
         ])),
```

```
        emailid: new FormControl("angular@gmail.com",Validators.compose([
          Validators.required,
          Validators.pattern("[^ @]*@[^ @]*")
        ])),
        password: new FormControl("abcd1234",this.validateLength)
      });
  }
  validateLength(formcontrol){
    if (formcontrol.value.length < 5) {
      return {"password" : true};
   }
  }
  value="welcome...:"
  onClickSubmit(data:Array<Data>){
    this.emailid=data["emailid"];
    console.log(data["uname"]);
    console.log(data["emailid"]);
    console.log(data["password"]);
}
}
```

App.module.ts

```
import { BrowserModule } from
'@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-
routing.module';
import { AppComponent } from './app.component';
import { ReactiveFormsModule } from
'@angular/forms';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    ReactiveFormsModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

App.component.html

```
<div>
        <form [formGroup]="formdata" (ngSubmit)
= "onClickSubmit(formdata.value)" >
            <input type="text"
class="fortextbox" name="USERNAME"
placeholder="username"
                formControlName="uname">
            <input type="text"
class="fortextbox" name="EMAILid"
placeholder="emailid"
                formControlName="emailid">
            <br/>
            <input type="password"
class="fortextbox" name="PASSWORD"
                placeholder="passwd"
formControlName="password">
            <br/>
            <input type="submit" [disabled] =
"!formdata.valid" class="forsubmit" value="Log
In">
        </form>
    </div>
    <p>
       Email entered is : {{emailid}}
    </p>
```

UseCase 15 – FormBuilder and ValueChange

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';

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

  constructor(private fb:FormBuilder) { }
  employeeForm:FormGroup;
  ngOnInit() {

  this.employeeForm=this.fb.group({
    name:['dummy',[Validators.required,Validators.minLength(2),Validators.maxLength(10)]],
    email:[''],
    skills:this.fb.group({
      skillName:[''],
      experienceInYears:[''],
      proficiency:['']
    }),
  });
  this.employeeForm.get("name").valueChanges.subscribe(value=>{
    //this.employeeForm.get('name').errors.minLength
      console.log("value changed....:"+value);
     // console.log("json value..:"+JSON.stringify(value));
  });
}
}
```

```
<div>
  <form [formGroup]="employeeForm"
           (ngSubmit) = "onClickSubmit(employeeForm.value)" >
      <input type="text" placeholder="name"
           formControlName="name">
     <input type="text" placeholder="email"
           formControlName="email">
     <br/>
     <div formGroupName="skills">
     <input type="text"  placeholder="skillname"
           formControlName="skillName">
     <br/>
     <input type="text"  placeholder="experienceInYears"
     formControlName="experienceInYears">
    <br/>
    <input type="radio" value="beginner"  name="proficiency"
    formControlName="proficiency">Beginner
    <input type="radio" value="intermediate"  name="proficiency"
    formControlName="proficiency">intermediate
    <input type="radio" value="advanced"  name="proficiency"
    formControlName="proficiency">Advanced
     </div>
     <input type="submit"
          [disabled] = "!employeeForm.valid"
```

```
          value="Log In">
  </form>
  <input type="button"
          (click) = "onLoadDataClick()"
          value="Load Data">
</div>
<p>
  Email entered is : {{email}}
</p>
```

UseCase 16 – Routing

Follow UseCase 8 and create a component, which can be linked via a url using Routing.

App.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { ChangeTextDirective } from './change-text.directive';
import { RouterModule } from '@angular/router';
import { NewCmpComponent } from './new-cmp/new-cmp.component';
@NgModule({
  declarations: [
    AppComponent,
    ChangeTextDirective,
    NewCmpComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    RouterModule.forRoot([
      {
         path: 'new-cmp',
         component: NewCmpComponent
      }
   ])
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

app.component.html

```
<a routerLink = "new-cmp">New component</a>

<br />
<br/>
<router-outlet></router-outlet>
```

Note: The <router-outlet> tag is the place the output of the component will get displayed.

UseCase 17 - Using Generics

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

@Component({
  selector: 'app-genericcomponent',
  templateUrl: './genericcomponent.component.html',
  styleUrls: ['./genericcomponent.component.css']
})
export class GenericcomponentComponent implements OnInit {
  water:Water;
  paint:Paint;
  paintbrush:PaintBrush<Paint>;
  waterbrush:PaintBrush<Water>;
  constructor() {
    this.water=new PlainWater();
    this.paint=new RedPaint();
    this.paintbrush=new PaintBrush<Paint>(this.paint);
    this.waterbrush=new PaintBrush<Water>(this.water);
  }

  execute():void{
    this.paintbrush.getT().doPaint();
    this.waterbrush.getT().doSprinkle();
  }
  ngOnInit() {
  }

}
class PaintBrush<T>{
  t:T;
  constructor(t:T){
    this.t=t;
  }
  getT(){
    return this.t;
  }
}
abstract class Paint{abstract doPaint():void;}
class RedPaint extends Paint{
  doPaint():void{
    console.log("red paint.....");
  }
}
abstract class Water{abstract doSprinkle():void;}
class PlainWater extends Water{
  doSprinkle():void{
```

```
    console.log("do sprinkle.......");
  }
}
```

UseCase 18 – Directive Creation

Directives in Angular is a js class, which is declared as @directive. We have 3 directives in Angular. The directives are listed below –

Component Directives

These form the main class having details of how the component should be processed, instantiated and used at runtime.

Structural Directives

A structure directive basically deals with manipulating the dom elements. Structural directives have a * sign before the directive. For example, *ngIfand *ngFor.

Attribute Directives

Attribute directives deal with changing the look and behavior of the dom element. You can create your own directives as shown below.

Now let us create a custom attribute directive by issuing the following command.

C:\SHOIAB\2019-STUDYMATERIAL\JS-FULLSTACK\ANGULAR\PROJECTS\firstproj>ng g directive changeText
CREATE src/app/change-text.directive.spec.ts (241 bytes)
CREATE src/app/change-text.directive.ts (149 bytes)
UPDATE src/app/app.module.ts (643 bytes)

App.module.ts will get updated automatically –

```
import { ChangeTextDirective } from './change-text.directive';
declarations: [
    AppComponent,
    ChangeTextDirective
  ],
```

change-text.directive.ts

```
import { Directive, ElementRef } from '@angular/core';
@Directive({
  selector: '[appChangeText]'
})
export class ChangeTextDirective {

  constructor(Element: ElementRef) {
```

```
    console.log(Element);
    Element.nativeElement.innerText="Text is changed by changeText Directive. ";
 }
}
```

App.component.html

```
    <div style="text-align:center">
            <span appChangeText >Welcome to {{title}}.</span>
    </div>
```

UseCase 19- Angular – Testing – Jasmine and Karma

jasmine - its a framework, which helps us to create our test

karma- it is a task runner and browse our test reports in browser.

TestBed-

UseCase 20 - ServiceWorker / PWA

App.component.html

```
<h1>My Posts</h1>
<app-post *ngFor="let post of posts" [content]="post.body" [title]="post.title"></app-post>
```

App.component.ts

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

import { Post } from './post.model';

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

  constructor(private http: HttpClient) {}
```

```
  ngOnInit() {
    this.http
      .get<Post[]>('https://jsonplaceholder.typicode.com/posts')
      .subscribe(fetchedPosts => (this.posts = fetchedPosts));
  }
}
```

App.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { HttpClientModule } from '@angular/common/http';

import { AppComponent } from './app.component';
import { PostComponent } from './post/post.component';
import { ServiceWorkerModule } from '@angular/service-worker';
import { environment } from '../environments/environment';

@NgModule({
  declarations: [AppComponent, PostComponent],
  imports: [BrowserModule, HttpClientModule, ServiceWorkerModule.register('ngsw-
worker.js', { enabled: environment.production })],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Post.model.ts

```
export interface Post {
    title: string;
    body: string;
    id: number;
    userId: number;
  }
```

>ng g component post

Post.component.css

```
article {
    padding: 1rem;
    margin: 1rem auto;
    box-shadow: 1px 1px 5px rgba(0, 0, 0, 0.5);
    font-family: 'Oswald', sans-serif;
```

```
    width: 30rem;
    max-width: 80%;
  }

  h1 {
    font-family: inherit;
    font-weight: bold;
  }
```

Post.component.html

```
<article>
  <h1>{{ title }}</h1>
  <p>{{ content }}</p>
</article>
```

Post.component.ts

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

@Component({
  selector: 'app-post',
  templateUrl: './post.component.html',
  styleUrls: ['./post.component.css']
})
export class PostComponent {
  @Input() title: string;
  @Input() content: string;
}
```

ng add @angular/pwa
ng build --prod
npm install -g http-server
dist/angular-pwa/http-server -p 8081
http://localhost:8081/index.html

Ngsw-config.json

```
{
  "index": "/index.html",
  "assetGroups": [{
    "name": "app",
    "installMode": "prefetch",
```

```
    "resources": {
      "files": [
        "/favicon.ico",
        "/index.html",
        "/*.css",
        "/*.js"
      ],
      "urls": [
        "https://fonts.googleapis.com/css?family=Oswald:300,700",
        "https://fonts.gstatic.com/**"
      ]
    }
  }, {
    "name": "assets",
    "installMode": "lazy",
    "updateMode": "prefetch",
    "resources": {
      "files": [
        "/assets/**"
      ]
    }
  }],
  "dataGroups": [
    {
      "name": "posts",
      "urls": [
        "https://jsonplaceholder.typicode.com/posts"
      ],
      "cacheConfig": {
        "maxSize": 5,
        "maxAge": "6h",
        "timeout": "10s",
        "strategy": "freshness"
      }
    }
  ]
}
```

UseCase 21 – Services and Dependency Injection

We might come across a situation where we need some code or service to be used everywhere on the page. It can be for data connection that needs to be shared across components, etc. Services help us achieve that. With services, we can access methods and properties across other components in the entire project.

To create a service, we need to make use of the command line. The command for the same is –

>ng g service <<service-name>> or create a file by name <<service-name>>.service.ts

MyService.service.ts

```
import { Injectable } from '@angular/core';
@Injectable()
export class MyService {
   constructor() { }
   showTodayDate() {
      let ndate = new Date();
      return ndate;
   }
}
```

App.module.ts

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

App.component.ts

```
import { Component, OnInit} from '@angular/core';
@Component({
  selector: 'myfirsttag',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit
{
constructor(private myservice:MyService){
    console.log("constructor called....");
  }
  todaydate;// = new Date();
  ngOnInit(){
    console.log("ng on init called.....");
    this.todaydate=this.myservice.showTodayDate();
  }
}
```

Note: when a component is in need of a service, then declare the service in two places.

1. **In the module.ts – providers section**
2. **In the constructor of the component**
   Angular understands and does injection of the service during runtime automatically.
3. **To use ngOnInit – do the following**
   a. **Declare in import statement.**

UseCase 22 – HttpService

Run the below command before working on Http Service

npm install rxjs@6 rxjs-compat@6 –save

App.component.ts

```
import { Component,OnInit } from '@angular/core';
import { MyService } from './MyService.service';
import { HttpClient } from '@angular/common/http';
import 'rxjs/add/operator/catch';
import 'rxjs/add/observable/throw';

interface EmpType{
  name:string;
  age:number;
  country:string;
}
@Component({
  selector: 'myfirsttag',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit{
  title = 'firstproj';
  isAvailable=true;
  constructor(private http:HttpClient){
    console.log("constructor called....");
  }
  mydata;
  ngOnInit(){
    console.log("ng on init called.....");
    this.mydata=this.http.get("assets/employee.json");
    this.mydata.subscribe((data:  Array<EmpType>) => {
     for(var i=0;i<data.length;i++){
      console.log(data[i].name);
     console.log(data[i].age);
     console.log(data[i].country);
    }
  });
  }
}
```

App.module.ts

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { SqrtPipe } from './app.sqrt';
import { MyService } from './MyService.service';
import { HttpClientModule, HTTP_INTERCEPTORS }
                              from '@angular/common/http';
@NgModule({
  declarations: [
    SqrtPipe,
```

Employee.json

Note: Save this file in the asset folder.

```
[
    {
      "name": "Douglas  Pace",
      "age": 35,
      "country": "MARS"
    },
    {
      "name": "Mcleod  Mueller",
      "age": 32,
      "country": "USA"
    }
]
```

```
    AppComponent
  ],
  imports: [
    HttpClientModule,
   BrowserModule,
    AppRoutingModule
  ],
  providers: [MyService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Note: We have declared an interface (A Datatype as per TypeScript standards to capture the incoming data from http request)

UseCase 23 – Observables

```
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
export class Student {
  id: Number;
  name: String;
  enrollmentnumber: Number;
  college: String;
  university: String;
}
@Injectable({
  providedIn: 'root'
})
export class ObservableserviceService {

  students: Student[] = [{
    id: 1,
    name: 'Krunal',
    enrollmentnumber:1212121121,
    college: 'VVP Engineering College',
    university: 'GTU'
},
{
    id: 2,
    name: 'Rushabh',
    enrollmentnumber: 110470116023,
    college: 'VVP Engineering College',
    university: 'GTU'
},
{
    id: 3,
    name: 'Ankit',
    enrollmentnumber: 110470116022,
    college: 'VVP Engineering College',
    university: 'GTU'
}];

  constructor() { }
  public getStudents(): any {
    const studentsObservable = new Observable(observer => {
           setTimeout(() => {
               observer.next(this.students);
           }, 1000);
    });

    return studentsObservable;
}
}
```

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

```
import { Student, ObservableserviceService } from '../observableservice.service';

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

  students: Student[] = [];

  constructor(private studentservice: ObservableserviceService) {}

  ngOnInit() {
      const studentsObservable = this.studentservice.getStudents();
      studentsObservable.subscribe((studentsData: Student[]) => {
          this.students = studentsData;
      });
  }
}
```

```
<div class="container">
    <div class="row" style="margin-top: 30px">
        <div class="col-md-3 col-xs-6" *ngFor="let student of students">
            <div class="card">
                <div class="card-body">
                    <h5 class="card-title">{{ student.name }}</h5>
                    <h6 class="card-subtitle">{{ student.enrollmentnumber }}</h6>
                    <p class="card-text">{{ student.college }}</p>
                    <p class="card-text">{{ student.university }}</p>
                    <a class="btn btn-primary" href="#" >Go somewhere</a>
                </div>
            </div>
        </div>
    </div>
</div>
```

APP.COMPONENT.HTML

```
<app-observablecomp></app-observablecomp>
```