**Angular 7**

**What is Angular?**

* Framework from google to build the client side applications
* Suitable to build the single page application(SPA)

**Why Angular ?**

* It supports modular approach so the application will have clear structure
* It has feature to build components which are reusable
* It supports the Validation and routing features which makes development quicker and easier
* Angular code are easily unit testable and maintainable
* It use typescript language from Microsoft

**Prerequisites:**

HTML,CSS ,JS , Basics of Typescript

**Installation:**

* Install node.js
* Install npm
* Install Angular CLI – it is an command line interface for angular. It generates the building blocks of the application.
* Install Text Editor – VS Code

**Angular:**

* It has one or more modules ( main module is called as RootModule or AppModule)
* Every Module is made up of components and services. The components are derived from the rootcomponent or AppComponent
* Each components has HTML and Class . HTML controls the view and class controls the logic of the view
* Services has the business logic of the application
* Modules interact with each other to render the view

**First Angular Application:**

Package.json -. List out the various modules which are dependencies to the angular application , which are gets installed under node\_modules

-template has some limitations

- cannot assign the value to the variable

- cannot access the global javascript variable ie. Window

**Attributes and Properties:**

* Attributes are HTML attributes.
* Properties are DOM properties

**Angular supports property binding:**

* It is useful to bind the Boolean property
* Property binding :- assigning the value to the property during runtime(dynamic value)

**ex:**

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

@Component({

selector: 'app-test',

template:`

<div>

From test component with name {{name}}

</div>

<h2>{{2+7}}</h2>

<h3> {{"Welcome"+name}}

<h5>{{name.length}} </h5>

<h5>{{greetUser()}} </h5>

<h5>{{siteUrl}} </h5>

<input id={{myId}} type="text" value="Vasantha" /> --property binding

<input [disabled]="isDisabled" [id]="myId" type="text" value="Vasantha" />

<input bind-disabled="isDisabled" [id]="myId" type="text" value="Vasantha" />

`,

styleUrls: ['./test.component.css']

})

export class TestComponent implements OnInit {

public name="Vasantha";

public siteUrl=window.location.href;

public myId="testId";

public isDisabled=true;

constructor() { }

ngOnInit() {

}

greetUser(){

return "Hello "+this.name;

}

}

**Class binding:**

Which allows dynamically add or remove the class to an html element based on the dynamic value.

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

import { isDefaultChangeDetectionStrategy } from '@angular/core/src/change\_detection/constants';

@Component({

selector: 'app-test',

template:`

<div>

From test component with name {{name}}

</div>

<h2 [class]="success">{{2+7}}</h2>

<h3 class="text-danger"> {{"Welcome"+name}} </h3>

<h5 >{{name.length}} </h5>

<h1 class="text-special">{{greetUser()}} </h1>

<h1 [ngClass]="messageClass"> Hi This is for class binding </h1>

<h5 [class.text-special]="hasError">{{siteUrl}} </h5>

<input id={{myId}} type="text" value="Vasantha" />

<input [disabled]="isDisabled" [id]="myId" type="text" value="Vasantha" />

<input bind-disabled="isDisabled" [id]="myId" type="text" value="Vasantha" />

`,

styles:[`

.text-success{

color:green;

}

.text-danger{

color:red;

}

.text-special{

font-style:italic;

}

`]

})

export class TestComponent implements OnInit {

public name="Vasantha";

public siteUrl=window.location.href;

public myId="testId";

public isDisabled=true;

public isSuccess=true;

public hasError=false;

public isDanger=false;

public isSpecial=true;

public messageClass={

"text-success":!this.hasError,

"text-danger":this.isDanger,

"text-special":this.isSpecial

}

constructor() { }

ngOnInit() {

}

greetUser(){

return "Hello "+this.name;

}

}

**Style binding:**

Based on the property value of the class styles are applied to the html element.

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

import { isDefaultChangeDetectionStrategy } from '@angular/core/src/change\_detection/constants';

@Component({

selector: 'app-test',

template:`

<div>

From test component with name {{name}}

</div>

<h1 [style.color]="hasError? 'red' : 'green'"> This is style binding </h1>

<h1 [ngStyle]="styleBindingClass"> This is style binding class </h1>

`,

styles:[`

.text-success{

color:green;

}

.text-danger{

color:red;

}

.text-special{

font-style:italic;

}

`]

})

export class TestComponent implements OnInit {

public name="Vasantha";

public siteUrl=window.location.href;

public myId="testId";

public isDisabled=true;

public isSuccess=true;

public hasError=true;

public isDanger=false;

public isSpecial=true;

public styleBinding="yellow";

public messageClass={

"text-success":!this.hasError,

"text-danger":this.isDanger,

"text-special":this.isSpecial

};

public styleBindingClass={

color:"blue",

fontStyle:"italic"

};

constructor() { }

ngOnInit() {

}

greetUser(){

return "Hello "+this.name;

}

}

One way( data binding ): bind the data from class property to html view.

Other way data binding: called Event binding and bind the value of the html to class property.

$event is special in angular which gives all the information about the event which has raised

Template reference variable: which is used to pass the value to the template class

<input #myInput type="text" name="test"/>

<button (click)="logMessage(myInput.value)">click me</button>

**Two way data binding :**

Binds data in both direction. From view to model and Model to view

Which enables model and view are in sync.

**Ex:**

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

import { isDefaultChangeDetectionStrategy } from '@angular/core/src/change\_detection/constants';

@Component({

selector: 'app-test',

template:`

<div>

From test component with name {{name}}

</div>

<input [(ngModel)]="name" type="text" name="test"/>

{{name}}

`,

styleUrls: ['./test.component.css']

})

export class TestComponent implements OnInit {

public name="";

constructor() { }

ngOnInit() {

}

logMessage(value){

console.log(value);

}

}

**Structural Directives:**

* Which are used to add or remove the html elements in dom

Which are,

* ngIf
* ngSwitch
* ngFor

ngIf and ngSwitch are used to render html elements conditionally

ngFor used to render the list of html elements

**ngif : Ex: 1**

<h2 \*ngIf="ifFlag else elseBlock"> Welocme to ngIf </h2>

<ng-template #elseBlock>

<h2> This is for else block </h2>

</ng-template>

**Ex:2**

<div \*ngIf="ifFlag; then thenBlock; else elseBlock"> </div>

<ng-template #thenBlock>

<h2> This is for if then condition </h2>

</ng-template>

<ng-template #elseBlock>

<h2> This is for else block </h2>

</ng-template>

**ngSwitch: Ex:**

<div [ngSwitch]="color">

<div \*ngSwitchCase="'red'"> you picked red color </div>

<div \*ngSwitchCase="'blue'"> you picked blue color </div>

<div \*ngSwitchCase="'green'"> you picked green color </div>

<div \*ngSwitchDefault> pick again </div>

</div>

`

,

styleUrls: ['./test.component.css']

})

export class TestComponent implements OnInit {

public color="black";

**ngFor Ex:1**

<div \*ngFor="let color of colors">

<h2> {{color}}</h2>

</div>

`

,

styleUrls: ['./test.component.css']

})

export class TestComponent implements OnInit {

public colors=["red","balck","green"];

let is an keyword in angular

Ex: 2 <div \*ngFor="let color of colors; index as i">

<h2> {{i}} {{color}}</h2>

</div>

Like index we have some other keywords which is used along with ngFor

First , last , odd , even

<div \*ngFor="let color of colors; even as i">

<h2> {{i}} {{color}}</h2>

</div>

<div \*ngFor="let color of colors; first as i">

<h2> {{i}} {{color}}</h2>

</div>

**Component Interaction:**

* parent to child interation happens by usingthe template reference variable
* child to parent interation happens using the events

**Pipes:**

* it allow us to transform data before displaying into view

Predefined pipes applicable for string properties

<h2> {{name | lowercase }}</h2>

<h2> {{name | uppercase }}</h2>

<h2> {{title | titlecase }}</h2>

<h2> {{name | slice:2:4 }}</h2>

**Number Pipes:**

<h2> {{5.689 | number:'1.2-3'}}</h2>

<h2> {{5.689 | number:'3.4-5'}}</h2>

<h2> {{5.689 | number:'3.1-2'}}</h2>

**Other pipes:**

<h2> {{0.25 | percent}}</h2>

<h2> {{0.25 | currency}}</h2>

<h2> {{0.25 | currency:'GBP'}}</h2>

<h2> {{0.50 | currency:'EUR':'code'}}</h2>

**Date pipes:**

<h2> {{date}}</h2>

<h2> {{date| date :'short'}}</h2>

<h2> {{date| date :'shortDate'}}</h2>

<h2> {{date| date :'shortTime'}}</h2>

<h2> {{date| date :'medium'}}</h2>

<h2> {{date| date :'mediumDate'}}</h2>

<h2> {{date| date :'mediumTime'}}</h2>

<h2> {{date| date :'long'}}</h2>

<h2> {{date| date :'longDate'}}</h2>

<h2> {{date| date :'longTime'}}</h2>

**Angular Services:**

* Service is a class with specific implementation

Useful to share data , implement application logic , connect to external source(database)

Naming convention .service.ts

**Dependency Injection**

* It’s a design pattern
* It’s a framework
* Angular has built in injector which acts as a container

**Http and Observables:**

* Make a http request from employee service
* Get the http response as a form of observable
* Transform this observable into array
* And assign this to local variable for later use in component