Angular Tutorial

To check node Version

Node -v

To check npm version

Npm -v

To check angular version

Ng --version

1. **Why we are using angular**
2. Angular is mobile support merits
3. All component based
4. Single page applications
5. Data biding automatically no need to refresh page
6. Performance is high
7. **What is single page applications**
8. No need to load css and js and bootstrap files again and again.
9. Whenever user interacts with app only change Its only replacing the contents only.
10. **What Is typescript and uses**
11. Using typescript we can transpile code to js.
12. Es6 is not support some browsers so we are using

Typescript to convert to es5.

1. **How to create angular application**
2. **Install angular cli**

npm install -g @angular/cli

**2) create workplace**

ng new appname

**3) Run application**

Cd appname

ng serve –open

**5) what is components**

Consists of three things

* + Template = user interface
  + Component Decoder = its have the metadata example selector

Import angular and use the component

Decoder.

* + class = its consists of data and method

**6) Nesting components**

First import file on module.ts

Second nested file directive want include the first comp.

**7) style component**

We can style component following methods:

1. In view template using style tag
2. Other wise using class inline style in html
3. Main html file we include the css file.
4. Otherwise we can specify in component stylesUrl
5. We can specify following method

Styles : [‘table{color:red}’]

**8) interpolation**

Interpolation is a binding data into view template.

Three types

1. One way data binding - class to ui= normal Template expressions
2. One way data binding - ui to class = event binding
3. Two way data binding

Class to Ui and ui to class

Example

[(ngModel)] = we need to import formsmodule in angular and use in input.

**9) Property binding**

<https://www.pluralsight.com/guides/attribute-class-style-bindings-angular>

We can binding the property values in property binding.

<span [innerHtml]= ‘firstname’></span>

**10) Attribute binding**

We can binding attribute following ways

<tr [attr.colspan]=”3”>Method</td>

**11) Class binding**

We can use class binding following ways

Method:1

<P [className]=”myclass”>hi</p>

Method:1.1

NgClass- Directive

<p [ngClass]=”myclass”></p>

Method:2

<p [class.myclass]=”isTrue”>hi</p>

<div [class.<className>]="condition"></div>

**12) Style binding**

We can use class binding following ways

Method:1

<P [style.color]=”red”>hi</p>

Method:2

<p [style.fontsize.px]=”isTrue:10:20”>hi</p>

Method : 3

<some-element [ngStyle]="{'font-size': '20px'}">Set Font size to 20px</some-element>

**13) event binding**

We can use class binding following ways

<input type = “text” (onclick)= “myfun1”>

Class

Myfun1()

{

Alert()

}

**14) Pipes**

Pipe is a converting data one form to another form.

**Example**:

* + - Lowercase
    - Uppercase

Using pipe must be followed by symbol **|**

<https://www.tutorialspoint.com/angular4/angular4_pipes.htm>

**15) Interface in Angular**

**Why and Uses**

This allows only you to check that the expected data received follows a particular structure

1. Using interface we can change strong data type.
2. Interface start with

Export interface Ifame()  
{  
code: string;

Name1: number

}

Now import that type into comp.

Also we set now

**Employee= ifame[];**

We can now give the data in employee.

1. This one public default one.
2. Interface is an not an part of angular.it is used to checking to the data type in angular.

**16) Service in angular**

<https://www.tutorialspoint.com/angular2/angular2_dependency_injection.htm>

Angular service is a provide data to reuse purpose.

We can use anywhere in component we are using it.

**Steps:**

1. ) we need import the DI.

2.Di is using for adds extra functionality to the class.

1. Here we are Importing from Injectable

**In service.js file**

import {

Injectable

} from '@angular/core';

@Injectable()

export class appService {

getApp(): string {

return "Hello world";

}

}

1. Now add the code on app.comp.ts

import {

Component

} from '@angular/core';

import {

appService

} from './app.service';

@Component({

selector: 'my-app',

template: '<div>{{value}}</div>',

providers: [appService]

})

export class AppComponent {

value: string = "";

constructor(private \_appService: appService) { }

ngOnInit(): void {

this.value = this.\_appService.getApp();

}

}

**17) Component Life cycle**

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

<https://www.tutorialspoint.com/angular2/angular2_lifecycle_hooks.htm>

<https://www.tutorialspoint.com/angular2/angular2_lifecycle_hooks.htm>

**Life cycle methods**

1. **ngOnchanges()**

this is used bound input current and previous values.

1. **ngOnInit()**

when compound initializes that time its execute. First data display data.

1. **ngDocheck()**

detect any changes in angular work or not that cheked by angular ngDocheck.

1. ngAfterContentinit();

II) ngAfterContentChecked

III) ngAfterViewinit();

iV) ngAfferviewchecked()

1. **ngOnDestroy()**

its destroy the component.

**Steps:**

**Step1:**

Import files on OnInit

Import { Onlint } from ‘@angular/core’;

**Step2:**

Next we want to add to class.

Export class HI **implements OnInit**{}

**Step 3:**

**Write life cycle**

**ngOnInit(){}.**

**18) Routing**

**This means navigate one place to another place.**

**Step1 ) Root Module file**

First import the **module.ts** following files

Import the all **navigation components** file in module.ts

**Import {RouterModule,Routes } from ‘@Angular/cli’;**

**Routes meaning consists of array of routes.**

**Const apps: Routes = [**

{path: ’home’, component: homecompont },

{path: ’home’, component: Appcompont },

**If not hit anything we will load following component**

{path:””, redirectTo: ‘/home’,pathmatch:’full’ },

**Page not found component following method**

{path: ‘\*\*’, component: pagenotComponet }

**];**

**We want tell to routes on main ngmodule.**

@NgModule({

Imports: [RouterModule.forRoot(apps)],

})

**Step 2) next we want add html file routerLink**

<a routerLink=’home’> Home</a>

<a routerLink=’app’>Apss</a>

<router-outlet>This one loads the routed component inside on this</router-outlet>

19) how to read data from json

<https://www.encodedna.com/angular/read-an-external-json-file-in-angular-4-and-convert-data-to-table.htm>

Step 1:

Module.js

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

step 2:

import on component following

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

import { HttpErrorResponse } from '@angular/common/https';

step: 3:

export class AppComponent {

constructor (private httpService: HttpClient) { }

arrBirds: string [];

ngOnInit () {

this.httpService.get('./assets/birds.json').subscribe(

data => {

this.arrBirds = data as string []; // FILL THE ARRAY WITH DATA.

// console.log(this.arrBirds[1]);

},

(err: HttpErrorResponse) => {

console.log (err.message);

}

);

}

}

20) what is Directives

Change the behaviour of Dom elements.

Change the appearance of dom elements.

Types:

1. Structural directive: **3 Types**

Modify the dom structure.

1. \*ngIf

It will execute true or false condition.

Also we can use <ng-template> then and else

<div \*ngIf=”true” then option1 else option2></div>

<ng-template #Option1><ng-template>

<ng-template #Option2><ng-template>

1. \*ngSwitchCase

First we used switchcase many case we are using this.

Ts.File

boo:string=”1”;

Html

<ul [ngSwitch]="boo">

<li \*ngSwitchCase="'1'">1</li>

<li \*ngSwitchCase="'2'">2</li>

<li \*ngSwitchCase="'3'">3</li>

<li \*ngSwitchCase="'4'">4 jjk</li>

<li \*ngSwitchDefault>Default</li>

</ul>

Shows : first li.

1. \*ngFor

\*ngfor using iterating purpose.

1. Attribute Directive

Modify the dom elements.

1. Attribute binding
2. Class Binding

**21) @Input()**

Parent data passing to child component is @input.

**Step:1**

Parent values bind into property binding

Valus:string= “Some values”;

Now we passing child component in html

<child [childPass]=” Valus”></child>

**Step :2**

In child component we getting the values

<p>{{childPass}}</p>

OverView

Decorder=> It cant visible example @component

Directive=> Attribute Directive, structural directive =>It can be visible

1. Directive type
2. **Component Decorder**

Always you have to mention **@component** in class.

And tell to browser this is my html page you have to render.

And you have add more metadata in @component

Directive.