Phase 2:

Day 1

14-08-2021

Node JS Overview

Babel

Webpack

Typescript :

Angular Framework

Html,CSS,JavaScript and Bootstrap

**DOM :** Document Object Model

Using DOM API we can read, write and update dynamically.

**Library and framework**

To read and write DOM properly.

jQuery

Coffee JS

Ext JS

Angular JS

Angular Framework

Backbone JS

React JS

Vue JS

D3 JS

Node JS

Node JS is not a library or framework. It is run time environment for JavaScript library or Framework.

JRE for Java

Node for JavaScript

Before Node JS

Before Node JS JavaScript is known as Client side scripting language.

**Frontend backend**

HTML,CSS,Js and Bootstrap Java

jQuery Spring boot

Asp.net

Php

Python

Node JS

After Node JS JavaScript can be use client side as well as server side scripting language.

In Node JS we can’t use document as well as window objects.

Node JS provide npm ie node package manager.

Using npm we can download external Node JS or JavaScript library or modules.

npm --version

syntax to download the externa dependencies.

npm install –g moduleName (globally)

Typescript : Typescript is open source scripting language which support all ES6 and Object oriented programming concept.

Typescript also known as super set of JavaScript.

Whenever we are going to write the code in typescript the file extension must be .ts. TS file can’t understand by browser or we can’t include ts file in html page. So we have to convert ts to js with help of transpiler.

Transpiler is a type of compiler which help to convert one format of file into another format of file.

Tsc : Typescript compiler.

**Babel**

JSX : converting JSX code into JavaScript

Converting ES6 to ES5 JavaScript code.

Typescript

demo.ts

console.log(“Welcome to Typescript”);

to convert ts to js

first convert ts to js

**tsc demo.ts**

after run

**node demo.js**

**Below command enable tsc transpiler**

npm install –g typescript globally

mac user

sudo npm install –g typescript

type script features.

1. Typescript support data types but JavaScript using ES5 or ES6 doesn’t support.

Syntax to create data types Typescript

var, let and const

let/var variableName:datatype;

let/var variableName:datatype=value;

**array concept in typescript**

In Typescript we can create array to store same type as well as different type of values to store.

Array methods

push

pop

Shift

unshift

splice

forEach

map

filter

find()

findIndex()

etc

typescript functions

// number of parameter must be match by default any types considers

function addNumber(a,b){

    console.log(a+b);

}

addNumber(10,20);

addNumber(10.10 , 20.30);

addNumber("Raj","Deep");

//addNumber(1);    // two parameter must be pass

//addNumber();       two parameter must be pass

// number of parameter and type of parameter must be match.

function sumOfNumbers(a:number,b:number){

        console.log(a+b);

}

sumOfNumbers(100,200);

sumOfNumbers(10.20,20.30);

//sumOfNumbers("Raj"," Deep")

// function with no return type , void means no return type.

function info() : void {

        console.log("welcome to info function")

        //return 100;

}

info();

// function with specific return type like string

function sayHello(name:string): string {

        return "Welcome "+name;

}

console.log(sayHello("Raj"))

// function with number return type

function add(a:number,b:number):number {

        let sum = a+b;

        return sum;

}

console.log(add(100,200))

// function with no return type as well as it can return any type.

// by default any consider.

function hello() : any {

    //return 100;

    //return "Welcome"

    return true;

}

console.log(hello());

let sumOfNumber = (a:number,b:number):number=>a+b;

console.log(sumOfNumber(100,200));

// function with optional parameter

// we can declare the variable with ?

// optional parameter declaration must be from right to left.

// we can make 0 or 1 or all varaible as optional parameter

// mandatory parameter must be left side in parameter.

// no gap between two optional paramter declaration.

// no mandatory mandatory parameter between two optional parameter.

// by default optional parameter value is undefined.

function empInfo(id?:number,name?:string,salary?:number,age?:number){

    console.log("id is "+id);

    console.log("name is "+name);

    console.log("salary is "+salary);

    console.log("age is "+age);

}

empInfo(100,"Raj",12000,21);

empInfo(101,"Ramesh",14000);    // age undefined

empInfo(102,"Raju");            // salary and age undefined

empInfo(103);                   // name, salary, age undefined

empInfo();                      //id,name,salary,age undefined

//default initialization

function studentInfo(sid:number=0,sname:string="Unknown",age:number=18): void {

    console.log("Sid is "+sid);

    console.log("Sname is "+sname);

    console.log("age is "+age)

}

studentInfo(1,"Raj",21);

studentInfo(2,"Ramesh");

studentInfo(3);

studentInfo();

// rest operator or parameter : this operator is use

// to receive 0 or 1 or many parameter.

// in one function we can declare only one parameter as rest parameter

// it must be last parameter insider function

function employeeDetails(id:number,name:string,salary:number,...skillset:string[]){

        console.log("id is "+id);

        console.log("name is"+name);

        console.log("salary is "+salary);

        console.log("skillset is "+skillset)

}

let skillInfo:Array<string>=["React","Angular","Node","MongoDB"];

employeeDetails(1,"Raj",12000,"C");

employeeDetails(2,"Raju",10000);

employeeDetails(3,"Ramesh",20000,"C","C++");

employeeDetails(4,"Ajay",25000,"HTML","CSS","JS","Angular");

employeeDetails(5,"Raju",35000,skillInfo[0],skillInfo[1],skillInfo[2],skillInfo[4]);

employeeDetails(6,"Raj",34000,...skillInfo);    // spread operator

// spread operator is use to pass the array value to rest operator.

Oops concept

We can create user-defined object using three way

Using function ES5 style

Using literal ES5 style

Using class ES6 style

object : any real world entity.

Properties or state -🡪 fields/variable.

Person

Behaviour --🡪 function / methods.

Place

Bank

Animal

Car

class   
  
  
constructor : constructor is a special type of function which help to create the memory.

To write constructor in typescript we have to use constructor keywords.

Constructor get call automatically when we create the object of that class.

Constructor no return type not even void also.

In the life of the object if you want to perform any task only one time that type of code we have to write inside constructor ex : initialization. In the life of the object if you want to perform any task more than one time that type of code we have to write inside a functions.

Phase 2:

Day 2

15-08-2021

Access specifiers

private and public

if variable or function is private we can’t call directly we well as through objects.

If variable or function is public we can call through object but not directly.

Normally other oops language we can use access specifier with instance variable or functions.

But typescript support access specifiers with constructor parameter variables.

class Employee {

   // private id:number;

    //private name:string;

    //private age:number;     // instance varaible

    constructor(private id:number,public name:string,private age:number){      // local variable

            //this.id = id;

            //this.name = name;

           // this.age = age;

    }

    dis():void {

        console.log("id is "+this.id);

        console.log("name is "+this.name);

        console.log("age is "+this.age);

    }

}

let emp1 = new Employee(100,"Raj",21);

emp1.dis();

Inheritance :

Inheritance is use to inherits properties and behaviour of old class to new class.

class OldClass { super class or base class or parent class

properties

behaviour

}

class newClass extends OldClass{ // child class or //derived class or sub //class

}

Interface :

Typescript support interface also.

Interface mainly use to provide the specification.

In typescript we can declare two types of interfaces.

1. Interface with only function
2. Interface with only properties.
3. Interface with properties and functions. Rarely we use.

Typescript doesn’t extends two classes at same time.

But it can implements more than one interfaces.

interface with only function is use to provide the specification.

interface Bank {

    withdraw():void;

    deposit(): void;

}

class Sbi implements Bank{

    withdraw(): void {

    }

    deposit() : void {

    }

}

class Hsbc implements Bank {

        withdraw(): void {

        }

        deposit(): void {

        }

}

Interface with properties.

This type of interface is use to create literal style object with same type of properties and their data type.

Modules : modules is collection or group of variable, functions, classes and interfaces.

When we developing big application we write the code in different files.

Both the files are connect using require(ES5 or JavaScript style) or import (ES6 or typescript) and export.

require, import and exports

module is like a package in java.

In typescript module it advisable we have to create the typescript configuration files.

Filename must be tsconfig.json : This file hold all typescript configuration details.

Syntax to create the tsconfig.json file

tsc –-init

after file create take the help of tsconfig.json file to convert all ts file to js file using command

tsc only

Webpack : Web pack is a static module bundler or bundle file. Web pack treat all files and assets as a modules. Web pack create dependencies graph that describe how modules are related to each other using the reference (require or import) and export.

So we can’t include js file which contains require statement.

We can’t include js file in html page. if file contains require, import and export.

So we have to create the webpack configuration file which contains configuration details.

webpack.config.js

module.exports={

    entry:"./main.js",

    mode:"development",

    output:{

        filename:"bundle.js"

    }

}

Now install two dependencies.

npm install –g webpack

npm install –g webpack-cli

or

npm install webpack

npm install webpack-cli

After install successfully please run webpack command to create the bundle file.

webpack

React JS is a library

jQuery is a library.

Angular is a open source framework belong to google.

Angular is use to create SPA (Single Page Application).

Library Vs Framework

Library use for only specific purpose. But framework use for multipurpose. Library doesn’t follow any standard rules and regulation. But Framework follow all rules and regulation.

Design pattern: Best practise or solution for repeating problem.

The implementation of all design pattern is taken care by framework.

Framework is heavy but library light weighted.

MVC : Angular is base upon the MVC framework. Model View controller.

React is View in MVC.

SPA : Single Page Application

Index.html welcome.html

Hyperlink

Submit button

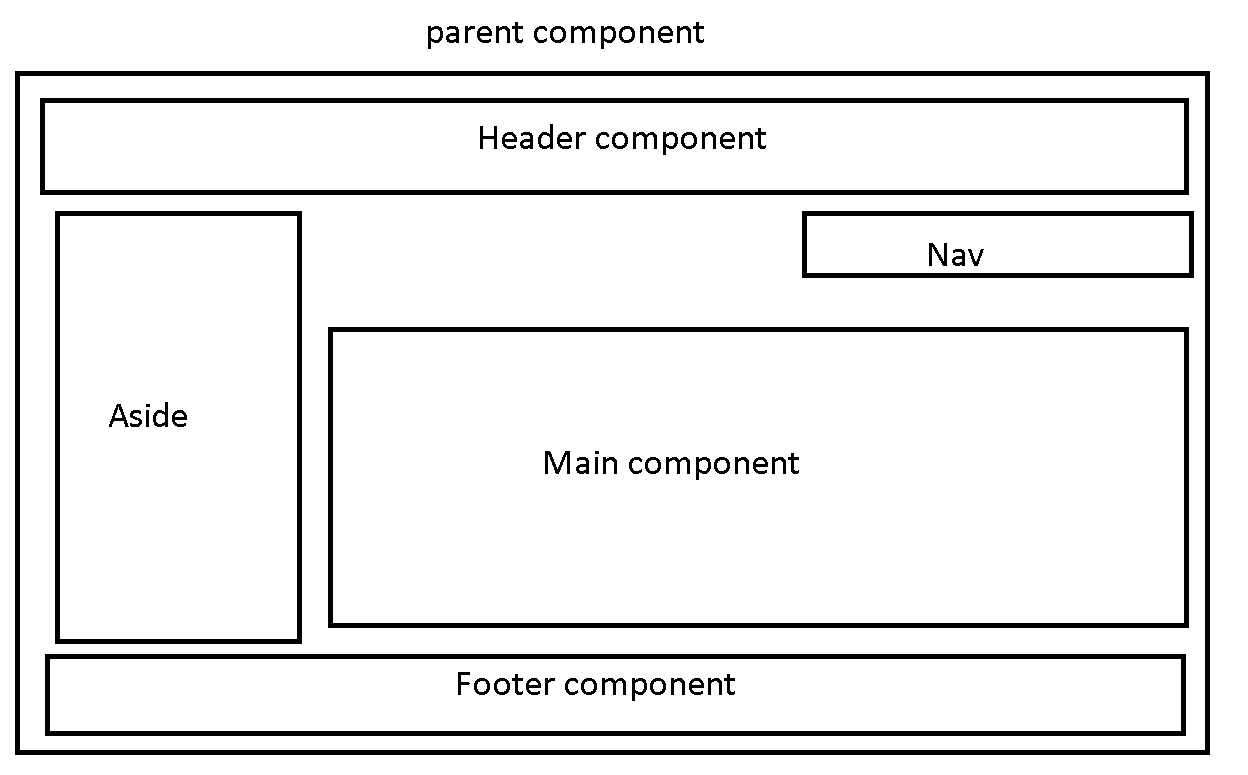
Normal button with JS code

When we move from one page to another page whole dom loaded in browser memory.

Using SPA we can load only part of the web page rather than whole page.

Angular and React JS use Component.

Component always control the view or part of the view of web page.



Angular JS 1.0 1.x

Html, css and Java ES5 and ES6

Angular Framework 2 to 12

Html, css, Typescript using ES6

Angular cli : command line interface

node –-version

npm -–version

ng command which help to create the angular project

next generation for DOM.

npm install –g @angular/cli

after executed

check

ng –-version

please create separate folder as angular projects

inside that project

**ng new project-name**

**like**

**ng new demo-app**

**routing -🡪 no**

**styling 🡪 css : enter**

**move inside a project folder using cd command**

**cd demo-app**

**to run the project we have to use the command as**

**ng serve**

**if it ask google policy y/no : you can enter y or n no issue.**

**after compiled 100% please open any browser and write the URL as**

[**http://localhost:4200**](http://localhost:4200)

Phase 2:

Day 3

21-08-2021

ng new project-name

routing 🡪 no

styling 🡪 css

after project created

please move inside a project folder

cd demo-app

ng serve (this command run inside project folder)

**if you get any error**

**npm install in project folder**

**after compiled 100% successfully**

**open the browser**

[**http://localhost:4200**](http://localhost:4200)

**open the project in VS code**

**then expand src folder**

**app folder**

**app.component.html 🡪 open this file.**

**app.component.ts -🡪 open this file**

**app.component.css 🡪**

**Angular Component 🡪 Component is use to control the view or part of view.**

**decorator : Decorator is a type of special function which help to add extra behaviour to class or property or function. Decorator is a part of typescript. Decorator is also known as meta-data. Meta-data means data about data.**

**Decorator is like a annotation in Java.**

**All decorator start with pre-fix @ followed by decorator name.**

**int a;**

**a: number;**

**@Component**

**@NgModule**

**@Injectable**

**@Pipe**

**@Input**

**@Output**

**@ViewChild**

**Etc**

**@Component**

**Selector**

**tempalteUrl**

**styleUrls**

**pre-defined attribute of component decorator.**

**Selector, templateUrl and styleUrl**

**in angular using component decorator we are creating user-defined tag with help of selector attribute.**

**<b>Welcome web page</b/>**

**<font color=”red”>Welcome</font>**

**Using selector attribute we are creating user-defined tag. Wherever we use user-defined tag the content which present in html page connect through tempalteUrl get replace.**

<app-root></app-root>

The code written in html page will display. The html page connected through templateUrl.

app.module.ts

module is a collection of more than one components.

@NgModule

ngModule contains set of predefined attribute

declaration : in this section we have to provide all component details.

imports : In this section we have to import pre-defined or user-defined modules. By default angular imported browser module. This module is use to display the output in browsing area.

providers : This attribute is use to provide angular service class details.

bootstrap : This property is use to load the parent component in web page or first page of our application.

app.module.ts-🡪 main module

login, customer and employee module are sub module

app.component.ts -🡪 Parent component

we can create more than one components.

Login module

Sign component

Singup component

Customer module

View all customer details component

Create customer info component

Employee module

Create employee component

Delete employee component

We will create own component.

Inside angular program folder (not insider demo-app)

Create new project with name angular-data-binding

ng new angular-data-binding

routing -🡪 no

styling -🡪 css

using cd command move inside angular-data-binding project

ng serve –o : This command compile the project after compiled successfully it will open in default browser with url <http://localhost:4200>

creating new component using ng command

ng generate component componentname

or

ng g c componentname

**Data binding**

Data binding is use to share the data between html (template) to component (ts file).

It provide the bridge which help to share to data between component to view and vice-versa(view to component).

2 types

1. **One way data binding** 
   1. **String interpolation** :

Component -----🡪view

{{}}

Example : we have to string interpolation in html or view page.

{{variableName}}

{{5+6}}

{{6-5}}

{{functionName()}}

* 1. **Property binding :**

Component ------🡪 View

[]

**In html**

**<input type=”text” value=”Raj”/>**

**<input type=”text” [value]=”variableName”/>**

**<p [innerText]=”lname”></p>**

**ng g c property-binding**

**String interpolation always all values string consider. But property binding apart string we can use other data types.**

**String interpolation we can use for only display purpose. Property binding we can do dynamic DOM operation with all data types.**

**Property binding concept we use in angular form validation.**

**<img src={{variableName}}/>**

**ng new types-of-directives**

**routing 🡪 no**

**styling 🡪 css**

**ng g c string-interpolation**

1. Two way data binding

Phase 2:

Day 3

21-08-2021

1. One way data binding
   1. String interpolation

Component ---🡪View

{{}}

* 1. Property binding

Component --🡪View

[]

* 1. Event binding :

View ----🡪Component

()

Angular use same event provided by JavaScript. Angular remove on pre-fix and event name wrap with ().

JavaScript Angular Event

onClick (click)

onDblClick (dblclick)

onMouseOver (mouseOver)

onSubmit (ngSubmit)

onChange (change)

etc etc

Using angular event we can call the typescript function and those function is part of component and we are calling without creating the object of that class.

Create new component using command as

ng g c event-binding

template reference :

Angular providing template reference concept which help to read the value of textfield, password field, radio button, checkbox in component.

Syntax to create the template reference

<input type=”text” #nameRef/>

1. Two way data binding

In Angular we can achieve two data binding using 2 ways

* 1. Event binding and property binding or string interpolation.

Event binding

View --🡪 component

String interpolation or property binding

Component --🡪View.

Both combine are two way data binding using some events.

* 1. We can achieve two way data binding using [(ngModel)]. If any changes happen in component it update on view and vice-versa.

Two way data binding syntax

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

ngModel is a pre-defined attribute part of FormsModule. So we have to import FormsModule in app.module.ts file in imports section.

**Types of directives**

Directive is a use to add extra behaviour to DOM elements.

Three types of directives

1. Component directive : Component is a type of directive which is use to create user-­defined tags with help of selector attribute. Selector ie like a user-defined tag. The code which written in html page replace in place of user-defined tags. The html code may be static or dynamic with data binding features.
2. Structure directive: using structure directive we can add or remove dom element ie html code.

\*ngIf and \*ngFor

Using structure directive we can use if statement and for loop.

1. Attribute directive : using attribute directive we can do styling for web page.

ngStyle it is equal to inline css

ngClass it is equal to class selector.

ng new types-of-directives

ng g c structure-directive

syntax for \*ngIf

<div \*ngIf="true">This is div tag</div>

\*ngFor is use to display the value of array or array object in template.

\*ngFor syntax

<div \*ngFor=”let variableName of arrayName”>

<p>{{variableName}}</p>

</div>