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