Angular 2.0 for JEE

Lesson 01 : TypeScript Fundamentals

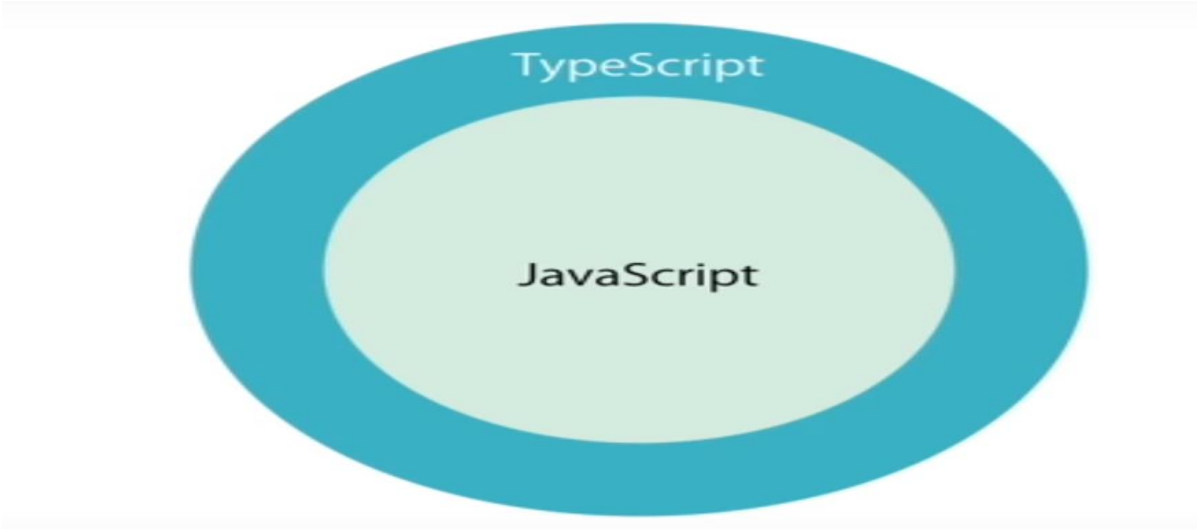
Capgemini

Lesson Objectives

- Introduction to Typescript
- JavaScript & Typescript
- The type system-Variable, Array
- Defining class and interface
- Arrow Functions

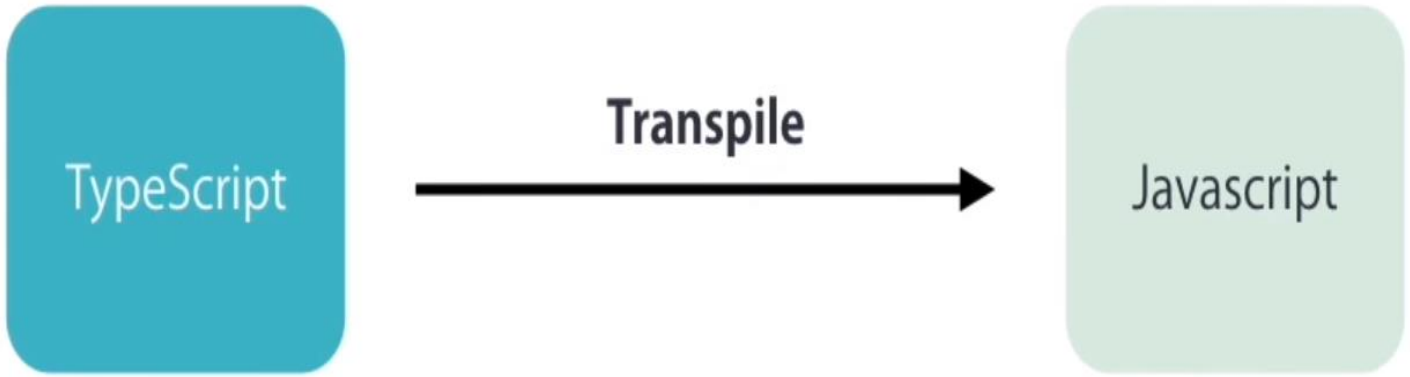
TypeScript

- **TypeScript** is an open-source programming language developed and maintained by Microsoft.
- It is superset of JavaScript.
- It is a strict syntactical superset of JavaScript, and adds optional static typing to the language.
- Anders Hejlsberg, lead architect of C# and creator of Delphi & Pascal, has worked on the development of TypeScript.
- TypeScript may be used to develop JavaScript applications for client-side or server-side -Node.js execution.



TypeScript

- Strong Typing
- Object Oriented features
- Compile time catching error
- Browser don’t understand typescript so we need to compile or trans pile into JavaScript. So Browser can understand it



Why TypeScript

- TypeScript can be used for cross browser development and is an open source project.
- Using TypeScript developers can apply class-based approach, compile them down to JavaScript without waiting for the next version of JavaScript.
- With TypeScript existing JavaScript code can be easily incorporated with popular JavaScript libraries like jQuery, Backbone, Angular and so on.
- Enable scalable application development with optional Static types, classes and modules. Static types completely disappear at runtime.
- TypeScript converts JavaScript Programming from loosely typed to strongly typed.
- JavaScript Version:
  - **ES5 (ECMAScript 5):** supported by all browsers
  - **ES6 (2015)**
  - **ES2016**
  - **ES2017**

Installing TypeScript

- First Install Node
  - Via npm (the Node.js package manager)--npm install -g typescript
  - Or DownLoad typescript compiler –master & set in class path & work
  - https://www.typescriptlang.org/play/ ---- work online
- Open Eclipse
  - Create Typescript project name as 'hello.ts'
  - Open command prompt & redirect to that eclipse folder
- For NPM users & use typescript without downloading
  - npm install -g typescript
  - At the command line, run the TypeScript compiler:
  - tsc hello.ts
  - When we write tsc hello.ts it will convert into js
  - Then write node hello.js

```
D:\AllDemoAngular\TypeScript>tsc hello.ts

D:\AllDemoAngular\TypeScript>node hello.js
hello world
```

Difference between let & var

- Using var

```
function doGet()
{
   for(var i = 0; i < 5; i++)
   {
      console.log(i);
   }
   console.log("Finally " + i);
}

doGet();
```

```
D:\AllDemoAngular\TypeScript>tsc diff.ts

D:\AllDemoAngular\TypeScript>node diff.js
0
1
2
3
4
Finally 5
```

- Now if you use 'let' instead of 'var' it will give compilation error, because scope is limited
- So in typescript we have to use 'let' instead of 'var'

Type Annotations

- Type annotations in TypeScript are lightweight ways to record the intended contract of the function or variable.

```
let empId: number;                                  --- number
let empName: string;                                --- string
let empFeedback: boolean;                           --- Boolean
let anyType: any;                                   --- any
let myArray: number[] = [1, 2, 3];                  --- number
let anyArrayType: any[] = [1, 'Rahul', false, true]; --- any array
```

Type Annotations

- Enum & Constant

```
const colored = 0;
const colorBlue = 1;
const colorGreen = 2;

enum Color { Red = 0, Green = 1, Blue = 2 };

let backgroundColor = Color.Red;

console.log(backgroundColor);
```

Type Assertion in TypeScript

- TypeScript allows changing a variable from one type to another.
- TypeScript refers to this process as *Type Assertion*.
- The syntax is to put the target type between < > symbols and place it in front of the variable or expression.

```
let str: string;

str.substring(2,3);


let str2;

(<string>str2).length;
(str2 as string).length;
```

Assertion in TypeScript

Arrow Functions

- => is a and also called a Arrow function

```
let log = function(message)
{
      console.log('Welcome to Arrow');
}

//Arrow function equivalent to above function
let doLog = (message) => console.log(message);

//Arrow function equivalent to no parameter  function
let withoutparameter = () => console.log();
```

Interfaces

- Interfaces plays many roles in TypeScript code. Its work same as other OOPs concept .

```
interface Employee{
    firstName: string;
    lastName: string;
    age: number;
    salary: number;
}

let employee: Employee={
        firstName: “Rahul”,
        lastName: “Vikash”,
        age: 32,
        salary: 1000
}

document.write(“Full Name is ” + this.employee.firstName + “ ” +
this.employee.lastName + “ Age is ” + this.employee.age + “<br />”);
```

Interface with array

```
interface Employee{
    firstName: string;
    lastName: string;
    age: number;
    salary: number;
}

//with array concept
let empArray: Employee[] = [];

empArray.push({
    firstName: “Abcd”,
    lastName: “Bcde”,
    age: 21,
    salary: 6000
});

document.write(“With array Full name is ” + this.empArray[0].firstName + “ ” +
this.empArray[0].lastName + “ Age is ” + this.empArray[0].age);
```

Function:

- Creating Functions

```
//2 parameter with number as return type
function getsum(numOne: number, numTwo: number): number{
    return numOne + numTwo;
}

let add = getsum(10,6);
document.write("Sum is " + add + "<br />");
```

```
//any number of data--know as rest parameter
function sumAll(...num: number[]){
    let sum: number = 0;
    for (let data of num) {
        sum = sum + data;
        document.write("Addition of number " + data + "<br />");
}
    document.write("Sum is " + sum + "<br />");
}

sumAll(6, 7, 8, 9);
```

Optional, Default

- ? Is know as optional parameter

```
//Optional parameter----? for optional & Default parameter
function doGet(one: number, two = 5, three?: number): void{
    //alert("hii");
    document.write(one.toString());
    document.write(two.toString());
    document.write(three.toString());
}

//doGet(10);
doGet(10);
```

Classes in TypeScript

- Traditional JavaScript focuses on functions and prototype-based inheritance, it is very difficult to built application using object-oriented approach.
- Starting with ECMAScript 6 (the next version of JavaScript), JavaScript programmers can build their applications using this object-oriented class-based approach.
- TypeScript supports public , private and protected access modifiers. Members of a class are public by default.

Classes in TypeScript (Contd…)

```
class Employee {
    empId: number;
    empName: string;
    empsalary: number;

    static emppf: number = 12;
    static company: string = 'CAPGEMINI';
}

let emp = new Employee();
emp.empId = 1001;
emp.empName = "Vikash";
emp.empsalary = 1111;
document.write("ID is " + emp.empId + " Name is " + emp.empName + "
company " + Employee.company);
```

Constructor -Typescript

```
class EmployeeOne {
    empId: number;
    empName: string;

    constructor(id: number, name: string) {
        this.empId = id;
        this.empName = name;
    }

    doGet(): void{
        document.write(this.empId + " " + this.empName);
    }
}

let empOne = new EmployeeOne(1001, "Abcd");
empOne.doGet();
```

Static Property

- In TypeScript we can also create static members of a class, those that are visible on the class itself rather than on the instances.

Static Property (Contd…)

```
class EmployeeOne {
    empId: number;
    empName: string;
    static numberOfEmployee: number = 0;

    constructor(id: number, name: string) {
        this.empId=id;
        this.empName=name;
        EmployeeOne.numberOfEmployee++;
    }

    doGet(): void{
        document.write(this.empId+" "+this.empName);
    }
    static getNumber(): number{
        return EmployeeOne.numberOfEmployee;
    }
}

let empOne=new EmployeeOne(1001, “Abcd”);
empOne.doGet();
EmployeeOne.getNumber();
```

Inheritance

- TypeScript allows us to extend existing classes to create new ones using inheritance.
- 'extends' keyword is used to create a subclass.
- 'super()' method is used to call the base constructor inside the sub class constructor.

Inheritance (Contd…)

```
class Animal {
    constructor(public name: string) { }
    move(distanceInMeters: number = 0) {
        console.log(`${this.name} moved ${distanceInMeters}m.`);
    }
}
class Snake extends Animal {
    constructor(name: string) { super(name); }
    move(distanceInMeters = 5) {
        console.log("Slithering...");
        super.move(distanceInMeters);
    }
}
class Horse extends Animal {
    constructor(name: string) { super(name); }
    move(distanceInMeters = 45) {
        console.log("Galloping...");
        super.move(distanceInMeters);
    }
```

Summary

- TypeScript is an open source project maintained by Microsoft.
- TypeScript generates plain JavaScript code which can be used with any browser.
- TypeScript offers many features of object oriented programming languages such as classes, interfaces, inheritance, overloading and modules, some of which are proposed features of ECMA Script 6.
- TypeScript is a promising language that can certainly help in writing neat code and organize JavaScript code making it more maintainable and extensible.
- Angular 2 is built in typescript

Demo

- TypeScript Demo

Lab

- Lab 1.1