TYPESCRIPT

TypeScript

- TypeScript is an open-source, object-oriented language developed and maintained by Microsoft, licensed under Apache 2 license.
- It is a superset of JavaScript, meaning that it contains all of the functionality of JavaScript and then some of it's own features. Therefore, any program written in valid JavaScript will also run as expected in TypeScript
- TypeScript compiles simply to plain JavaScript
- TypeScript offers us more control over our code via type annotations, interfaces, and classes
- TypeScript was created by Microsoft and was released in 2012 after two years of development. It was created to allow for optional static type checking, which would be particularly useful when developing large scale applications

TypeScript

- JavaScript is dynamically typed. Therefore, programs written in JavaScript do not know the data type of a variable until that variable is assigned a value at runtime. The variable can be reassigned or coerced into a value of a different type with no issues or warning. This can result in bugs that are often overlooked, especially in large applications.
- TypeScript, on the other hand, uses static typing. Variables can be given a type when they are declared. TypeScript will check types at compile time, and throw an error if the variable is ever given a value of a different type. However, the error does not prevent the code from executing. The code will still be compiled into plain JavaScript and run normally. In this way, TypeScript is kind of like a “spellcheck” for your code. It will let you know when something looks off, but it won’t change how your code runs.

Data Type and Variable declaration

- Like JavaScript and any other language, TypeScript also provides basic data types to handle numbers, strings, etc. Some common data types in TypeScript are: number, string, boolean, enum, void, null, undefined, any, never, Array and tuple.

Declaring a variable

- Variable can be declared using var ,let and const

```
var message:string = 'hello world'
console.log(message)



function display_message() : void{
    let message:string = "Hello World"
 console.log(message)
}
display_message()
```

Enum Types

```
enum Month{
    jan="January",
    feb="February",
    mar="March",
    apr="April",
    week=1

}
var monthType=Month.week
console.log(monthType)
```

Never type

```
var myfunction = function(): never{
while(true)
{
    console.log("never end")
}
}

myfunction()
```

Arrays

- var arrayname:datatype[]

- eg:-

```
var users:string[]
users=["admin","dba","developer"]
console.log(users[0])
```

Functions in typescript

- function function_name (param1[:type], param2[:type], param3[:type]):datatype
- {
- }

Rest Parameters

```
var myrestnum=function(...nums:number[]):void{
    var sum:number=0
    for(var i=0;i<nums.length;i++)
    {
     sum+=nums[i];
    }
    console.log(sum)
}
myrestnum(11,12,30,14)
```

Lambda Functions

- Lambda refers to anonymous functions in programming. Lambda functions are a concise mechanism to represent anonymous functions. These functions are also called as **Arrow functions**.

- ([param1, parma2,…param n])=>statement;

Lambda functions

- var add=(a:number,b:number)=>a+b

- var result:number = add(12,11)
- console.log(result)

Classes in TypeScript

```
class Person
{
    name:string;
    address:string

    constructor(name:string,address:string)
    {
 this.name=name
 this.address=address
    }
}

var employee=new Person("Peter","23 old street")
console.log(employee.name)
console.log(employee.address)
```

Classes and Inheritance

- TypeScript supports the concept of Inheritance. Inheritance is the ability of a program to create new classes from an existing class. The class that is extended to create newer classes is called the parent class/super class. The newly created classes are called the child/sub classes.

Inheritance

- class child_class_name extends parent_class_name

Access Modifiers

- Like others programming languages, TypeScript supports access modifiers at the class level. TypeScript supports three access modifiers - public, private, and protected.

1. **Public** - By default, members (properties and methods) of TypeScript class are public - so you don't need to prefix members with the public keyword. Public members are accessible everywhere without restrictions
2. **Private** - A private member cannot be accessed outside of its containing class. Private members can be accessed only within the class.
3. **Protected** - A protected member cannot be accessed outside of its containing class. Protected members can be accessed only within the class and by the instance of its sub/child class.