**Type Script Interface**

Interfaces define properties, methods, and events, which are the members of the interface.

An Interface is a structure which acts as a contract in our application. It defines the syntax for classes to follow

We cannot instantiate the interface, but it can be referenced by the class object that implements it.

The TypeScript compiler uses interface for type-checking (also known as "duck typing" or "structural subtyping") whether the object has a specific structure or not.

The interface contains only the declaration of the methods and fields, but not the implementation.

Interface Declaration

interface interface\_name {

// variables' declaration

// methods' declaration

}

An interface is a keyword which is used to declare a TypeScript Interface.

An interface\_name is the name of the interface.

An interface body contains variables and methods declarations.

interface testinter

{

id:number;

disp()void;

}

class testinter implements testinter

{

}

**Class & Interfaces**

interface IEmployee {

empCode: number;

name: string;

getSalary(empCode: number) :void;

}

class Employee11 implements IEmployee {

empCode: number;

name: string;

constructor(code: number, name: string) {

this.empCode = code;

this.name = name;

}

getSalary(empCode:number):void {

// return 20000;

console.log(2000);

}

}

let emp111 = new Employee11(1, "Steve");

console.log(emp111.getSalary(12))

**Example Interface and Objects**

interface IPerson {

firstName:string,

lastName:string,

sayHi: ()=>string

}

var customer:IPerson = {

firstName:"Rama",

lastName:"Krishna",

sayHi: ():string =>{return "Hi How Are U"}

}

console.log("Customer Object ")

console.log(customer.firstName)

console.log(customer.lastName)

console.log(customer.sayHi())

var employee:IPerson = {

firstName:"Vijay",

lastName:"Shankar",

sayHi: ():string =>{return "Hello!!!"}

}

console.log("Employee Object ")

console.log(employee.firstName);

console.log(employee.lastName);

**Extending interface**

interface Person {

name:string

age:number

}

interface Employee extends Person {

gender:string

empCode:number

}

let empObject = <Employee>{};

empObject.name = "Ram Laxman"

empObject.age = 23

empObject.gender = "Male"

empObject.empCode = 443

console.log("Name: "+empObject.name);

console.log("Employee Code: "+empObject.empCode);

Extending more than one interface

-------------------------------

interface a

{

id:number

}

interface b

{

nm:string

}

interface c extends a,b

{

addr:string

}

let d = <c>{};

d.id=456

d.string="ravi"

d.addr="tnhb"

console.log(d.id)

**Index Type Interface**

Indexable types have an index signature that describes the types we can use to index into the object, along with the corresponding return types when indexing.

interface nameArray {

[index:number]:string

}

let myNames: nameArray;

myNames = ['ram', 'laxman', 'sita'];

console.log(mynames[2])

**Difference between interface and inheritance**

**Interface**

An Interface is a structure which acts as a contract in our application. It defines the required functions, and the class is responsible for implementing it to meet that contract.

In an interface, we can only declare properties and methods.

An interface type objects cannot declare any new methods or variables.

Interface enforces the variables and methods which have to be present in an object.

Interface are classes that contain body-less structure (abstract or virtual functions). So, we have to derive the interface and then implement all of the functions in the subclass.

**Inheritance**

Inheritance is object-oriented programming that allows similar objects to inherit the functionality and data from each other.

In inheritance, we can use a superclass to declare and defines variables and methods.

In this, we can declare and define its own variables and methods of a subclass that inherits a superclass.

A subclass extends the capability of a superclass to suit the "custom" needs.

Inheritance is the process where one subclass acquires the properties of its superclass.