**TYPESCRIPT**

1. **CLASSES:**

**Example 1:**

class Square

{

side:number; //instance variable

area() : number //method

{

return (this.side \* this.side);

}

}

let ob = new Square(); //Creating an Object

ob.side = 10; //Accessing instance variable & methods

console.log("Area of a Square is:"+ob.area());

**Example 2:**

class Box

{

width:number;

depth:number;

height:number;

constructor(x:number, y:number, z:number) //constructor

{

this.width = x;

this.depth = y;

this.height = z;

}

volume() : number

{

return (this.width \* this.depth \* this.height);

}

}

let ob = new Box(5,10,15);

console.log("Volume of the box:"+ob.volume());

**Example 3:**

class Rectangle

{

len:number;

bre:number;

constructor(...rest)

{

this.len = rest[0];

this.bre = rest[1];

}

display() : void

{

console.log("Value of len:"+this.len);

console.log("Value of bre:"+this.bre);

}

area(): number

{

return (this.len \* this.bre);

}

}

let ob = new Rectangle(2,3);

ob.display();

console.log("Area="+ob.area());

1. **any datatype:**

let r : any;

r = 10;

r = "hello";

r = true;

r = () => "Hello World"; //arrow function

1. **INTERFACES:**

**Example 1:**

interface Box

{

width:number;

depth:number;

height:number;

volume():number;

}

let ob:Box = {

width:10,

depth:15,

height:12,

volume: function()

{

return this.width\*this.depth\*this.height;

}

}

console.log("Volume of the Box:"+ob.volume());

**Example 2:**

interface Person

{

firstname:string;

lastname:string;

age:number;

getFullName():string;

getAge():number;

}

//class implementing an interface

class Employee implements Person

{

firstname:string;

lastname:string;

age:number;

getFullName():string

{

return this.firstname+" "+this.lastname;

}

getAge():number

{

return this.age;

}

constructor(fn:string,ln:string,a:number)

{

this.firstname = fn;

this.lastname = ln;

this.age = a;

}

}

let ob = new Employee("CVR","CSIT",20);

console.log(ob.getFullName());

console.log(ob.getAge());

1. **GENERICS:**

* **GENERIC FUNCTIONS:**

**Example 1:**

function identity<T>(arg1:T) : T

{

return arg1;

}

let ob1 = identity<string>("Hello CSIT");

let ob2 = identity<number>(120);

let ob3 = identity<boolean>(true);

console.log(ob1);

console.log(ob2);

console.log(ob3);

**Example 2:**

function myfun<t,u,v>(arg1:t,arg2:u,arg3:v) : v

{

console.log(arg1);

console.log(arg2);

return arg3;

}

let val = myfun<number,string,boolean>(12,"hello",false);

console.log(val);

* **GENERIC CLASSES**:

class StudentInfo<T,U,V>

{

id: T;

name : U;

branch: V;

constructor(a:T, b:U, c:V)

{

this.id = a;

this.name = b;

this.branch = c;

}

displayDetails() : void

{

console.log("Id:"+this.id);

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

console.log("Branch:"+this.branch);

}

}

let ob = new StudentInfo<number,string,string>(301,"CVR","CSIT");

ob.displayDetails();

let ob2 = new StudentInfo<string,string,string>("19481A3301","CVR","CSIT");

ob2.displayDetails();

1. **DECORATORS:**

**Class Decorator:**

@simpleDecorator

class Bug

{

title: string;

constructor(s:string)

{

this.title = s;

}

}

function simpleDecorator(constructor: Function)

{

console.log("Simple Decorator is called");

}

**Decorator Factory:**

@factory("Hello")

class Demo

{

//class members

}

function factory(n:string)

{

return function(constructor: Function)

{

console.log("Parameter passed:"+n);

};

}

**Generalized example for all the decorator types:**

**@classDecorator**

class Person {

**@propertyDecorator**

public name: string;

**@accessorDecorator**

get fullName() {

// ...

}

**@methodDecorator**

printName(**@parameterDecorator** param: string) {

// ...

}

}