npm install -g typescript // install typescript

tsc -v // check typescript version

node <filename.js || filename.ts> // check file output from cmd

tsc <filename.ts> // create a .js file for the .ts file

tsc –init //configure typescript to the folder

tsc // convert all .ts file of rootDir to outDir as .js file

->data types of Typescript are -> Boolean, number, string, array(number, character, any), tuple, enum, unknown, any, union

->Access Modifier- Public, Private, Protected

///////////////////////////////////////////////// CODE //////////////////////////////////////////////////

//string, number, boolean

let fname: string="Saif";

let lname: string="Uddin";

//console.log(`Hi! I am ${fname} ${lname}`);

//array(number, character, any)

let list: number[]=[8,4,3];

//console.log(list[0]);

//tuple

let x:[string,number][];

x=[

    ['Hello1',100],

    ['Hello2',200],

    ['Hello3',300]

];

//console.log(x[1][1]);

//enum

enum Color{

    Red=2, //Default start from 0

    Green,

    Blue=9,

    White

}

let y:string =Color[9];

//console.log(Color.Red);

//console.log(Color.Green);

//console.log(Color.Blue);

//console.log(Color.White);

//console.log(y);

//unknown

let notsure:unknown=true;

//console.log(typeof(notsure));

//any

let looslytype:any=4;

let looslytype1:any="Saif";

//console.log(looslytype);

//console.log(looslytype1);

//union

let a: number|string=7;

//console.log(a);

// if else

if(.5){

    //console.log('Yes');

}else{

    //console.log('No');

}

//loop

let items1:number[]=[55,44,33];

for(let i of items1){

    //console.log(i);

}

let i=0;

while(items1[i]){

    //console.log(items1[i]);

    i++;

}

let j=0;

do{

    //console.log(items1[j]);

    j++;

}

while(items1[j])

//object

type user={

    username:string,

    password:number

}

interface user1{

    username:string,

    password:number

}

const User:user1={

    username:'saif',

    password:12345

}

//console.log(User.username);

//console.log(User.password);

//Type Assertion

let m:unknown='sa'

let n:number=(m as number);

let o:number=<number>m;

//console.log(o);

//function

function sum(x:number,y:number):number{

    return x+y;

}

let p:number=sum(200,300);

//console.log(p);

//alert(p);

//function without return

function sum2():void{

    //console.log(100);

}

sum2();

function sum3(x:number,y:number):void{

    //console.log(x+y);

}

sum3(100,200);

function printLabel(myObj:{label:string,size:number}){

    //console.log(myObj.label);

    //console.log(myObj.size);

}

let myObj={label:'Saif',size:10};

printLabel(myObj);

interface LabelSize{

    label:string,

    size:number

}

function printLabelInterface(myObj:LabelSize){

    //console.log(myObj.label);

    //console.log(myObj.size);

}

printLabelInterface(myObj);

//interface

interface Point{

    x:number,

    y:number

}

let p1:Point={x:10,y:20};

//console.log(p1.x);

//interface with readonly properties

interface Point2{

    readonly x:number,

    readonly y:number

}

let p2:Point2={x:10,y:20};

//p2.x=30;//error because can't change readonly value

//console.log(p2.x);

//function in interface

interface Calculate{

    (x:number,y:number):number

}

let obj:Calculate=(x:number,y:number):number=>{

return x+y;

}

let sum4:number=obj(5,10);

//console.log(sum4);

// class

class Greeter{

    Greeting! : string;

    constructor(msg:string){

        this.Greeting=msg;

    }

}

let objG=new Greeter("Hello");

//console.log(objG.Greeting);

// class with getter setter

class Greeter2{

    Greeting! : string;// can use public/private/protected here

    constructor(){

    }

    set Greetings(gtr:string){

        this.Greeting=gtr;

    }

    get Greetings():string{

        return this.Greeting

    }

}

let objG2=new Greeter2();

objG2.Greetings="Hellow World";

//console.log(objG2.Greeting);

//class implements interface

interface ClockInterface{

    currentTime:Date;

    sayTime():string;

}

class Clock implements ClockInterface{

    currentTime: Date=new Date();

    sayTime():string{

        return `Current Time Is : ${this.currentTime}`

    }

    constructor(){}

}

let objClock= new Clock();

//console.log(objClock.sayTime());

//inheritance

class P{

   x!:string;

   constructor(a:string){

this.x=a;

   }

   say(){

    console.log("Say Parent");

   }

}

class Q extends P{

    y!:string;

    constructor(a:string,b:string){

        super(a)

        this.y=b

    }

    say(){

        console.log("Say Child");

       }

}

let p3=new P("Hello")

let q=new Q("Hello","World")

//console.log(p3.say());

//generic

function identity<Type>(arg:Type):Type{

return arg;

}

//console.log(identity(20));

//console.log(identity("hello"));

//generic with array

function identityA<Type>(arg:Type[]):Type[]{

    return arg;

    }

    let str=identityA(['AA','BB','CC'])

    str.push('DD');

    //console.log(identityA([20,30,40]));

    //console.log(identityA(['A','B','C']));

    //console.log(str);