**FUNCTIONS IN TYPESCRIPT**

* Typescript functions are similar to JavaScript functions.
* Most of the time we don’t have to define a return type for them, due to type inference.
* The function that doesn't return anything has a **void** return type.

e.g.: 1. Function with explicit return type definition.

function printName(name:string){

console.log(name)

}

2. Function utilizing the type inference.

function addition(input1:number, input2:number) {

return input1 + input2

}

**FUNCTION AS TYPE**

* The **Function** keyword is used to define that a variable can hold any type of function.
* It has first letter as uppercase.

E.g.

let testFunction:Function;

testFunction = () => {}

* If we want to be more specific about which type of function a variable can hold we can define its type as:

let testFunction:(input:number)=>number;

testFunction = (input:string)=> input //Error

testFunction = (input:number)=> input //Success

**CALLBACK FUNCTIONS**

* We can define the type of a callback function as:

const printName = (firstName:string, lastName:string, print:(result:string)=>void) =>{

const result = `${firstName} ${lastName}`;

print(result)

}

printName(‘Harshit’,’Bhawsar’,(result)=> console.log(result))

**FUNCTION RETURN TYPE**

The function’s return type can be specified as :

* If a function returns nothing its return type will be **void.**
* If a function is expected to crash the code execution its return type will be **never.**

function add(input1:number, input2:number): number {

return input1 + input 2

}

function fiveStar(message:string):void {

console.log(message)

}

function thrownError(message:string, code:number):never {

throw {message, code};

}