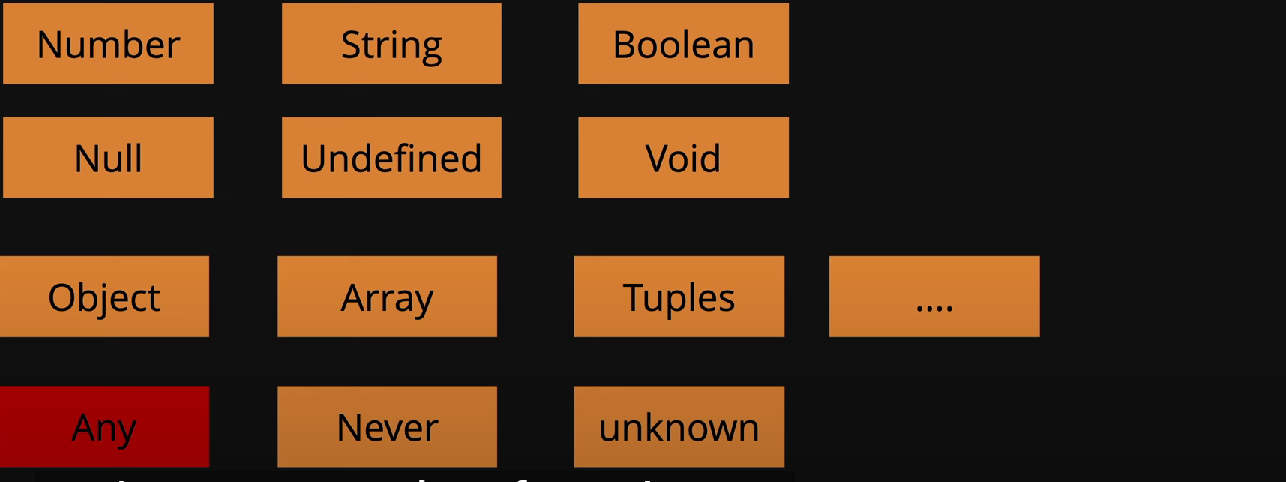
**TYPESCRIPT**

Typescript is the superset of javascript. What does typescript does: Static checking. It basically analyze the code as we type. Typescript is not about reinventing javascript. Typescript is all about type safety.

We write all the code in the form of **.tsx** form. We write our typescript code which then transpiled (converted) into javascript. Typescript is a development tool. It is just the layer above the javascript. Our code/project still runs in javascript. Typescript helps us to write better code with lesser problems with more scalable and maintainable code that is easily understandable and chances of producing error is very less but it is by no means that is a standalone language in itself. It is like a wrapper around a javascript.

Npx stands for node package executer.

## *28-December:*

Types available in typescript is 

Here using any means we are intentionally making our code more vulnerable or more like a javascript. This is how we declare the variable in typescript

let greetings: string = "Hello Manpreet Singh Goindi";

typeScript is smart enough to get all the values. Typescript itself tells us what to use or what not to use like if a variable is assigned a value to be of string and rather than writing string we are writing an integer, it will give us error moreover when we hover on our variable it will also show us the type it is of.

Where to use **ANY** :

In TypeScript, **any** type ko tab use karte hain jab hume kisi variable ka exact type pata na ho ya jab hume ek flexible aur generic type chahiye jo kisi bhi type ka data store kar sake. any ko samajhne ke liye kuch key points hain:

### ****1. Jab koi type ka pata nahi ho****

Agar koi value ka type runtime par decide hota ho ya hume exact type ka idea nahi ho, to hum any use karte hain.

typescript

Copy code

let randomValue: any;

randomValue = 42; // number

randomValue = "hello"; // string

randomValue = true; // boolean

### ****2. External libraries ke saath****

Agar aap kisi external library ko use kar rahe ho jo TypeScript types ko support nahi karti, to temporary solution ke liye any use hota hai.

typescript

Copy code

import someLibrary from 'unknown-library';

const result: any = someLibrary.doSomething();

### ****3. Gradual typing ke liye (Code migration)****

Agar JavaScript code ko TypeScript me migrate kar rahe ho aur har variable ka type define karna possible nahi ho, to any ek transitional type ke taur par use hota hai.

typescript

Copy code

function processInput(input: any) {

console.log(input);

}

### ****4. Avoid karne ke reasons****

* any TypeScript ki type safety ko bypass karta hai, jo future bugs ko lead kar sakta hai.
* Code readability aur maintainability kam ho jati hai.
* Alternatives like unknown aur specific types use karna better hota hai.

typescript

Copy code

let something: unknown; // better alternative to `any`

### Conclusion:

any ko sirf tabhi use karein jab koi aur option na ho ya quick fixes ke liye. Long-term ke liye, strongly typed code likhne ki aadat dalni chahiye.

It basically hen you don’t want any particular value to cause typechecking errors.

## *29-December: functions in typescript*

The whole idea of typescript is to make less mistakes regarding to the type we are using. Here in this function we are allowed to use nums.toUpperCase with the value where we only need numbers, this function also takes string as an input which is not correct as we are using number as our data-type.

function addTwo(num: number){

// function addTwo(num){

    // nums.toUpperCase;

    return num +2;

}

addTwo(5);

export{}

in-order to solve this we use number with num which is passed as parameter in the addTwo function name, with this we don’t need to check the function type.

function addTwo(num: number){

// function addTwo(num){

    // nums.toUpperCase;

    return num +2;

}

function getUpper(val: String){

    return val.toUpperCase();

}

function signUpUser(name: String, email: String, isPaid: boolean){

}

let loginUser = (name: String, email: String, isPaid: boolean = false) => {}

loginUser("Manpreet", "man@gmail.com");

signUpUser("Manpreet", "man@gmail.com", false)

addTwo(5);

getUpper("Manpreet");

export{}

Here we are having the loginUser where we can see how to set default values the default value syntax is like isPaid :Boolean = false, this way we can set default Boolean value or any value by default.

typescript is smart enough as when we change the context in our variable typescript itself picks the data-type and change its value/data-type accordingly. Like when I changed my value of hero from string to integer the value also itself changes.

// const heros = ['thor', 'spiderman','ironman']

const heros = [1, 2, 3]

heros.map(hero => {

    return `hero is ${hero}`

})

Whenever we add void it means that our function will not return any value.

Never type represents the values which are never observed. In a return type, this means that the function throws an exception or terminates execution of the program. Never also appears when typescript determines there’s nothing left in a union.