**What is TypeScript ?**

TypeScript is a superset of JavaScript in which you add types to JavaScript. JavaScript is a loosely typed language which leads to lot of types error in production code. With TypeScript developers can catch those error, even before running the code.

**What is the main difference between TypeScript and JavaScript?**

This is the core difference between JavaScript and TypeScript. While JavaScript is dynamically-typed, TypeScript is a statically-typed superset of JavaScript, which means it offers strict static typing as an option but will allow dynamic typing, as well.

**What is explicit and implicit type assignment?**

**Explicit/ Annotations** means writing out the type. Like below -

**let firstName: string = "Nabendu";**

**Implicit/ Inference** means TypeScript will guess the type, based on the value. Like below type will be considered a number

**let age = 41;**

**Difference between any, unknown and never in TypeScript?**

The type of any is used to assign any type of a variable. It will not give error even if you reassign another type. Like below -

**let x: any = 10;**

**x = 'hello'; // No TypeScript error**

**console.log(x.toUpperCase()); // No TypeScript error**

The type unknown is better than type any, because it requires us checking the type before performing operations on value. Like below -

**let y: unknown = 10;**

**// Type assertion needed before using y as number**

**if (typeof y === 'number') {**

**console.log(y.toFixed(2));**

**}**

The type never represents value that never occurs. It is typically used for return statements of function that doesn’t returns properly. Like below -

**function throwError(message: string): never {**

**throw new Error(message);**

**}**

**Class and Constructor in JavaScript and TypeScript**

In JavaScript and TypeScript, classes and constructors provide a structured way to create objects using the object-oriented programming (OOP) paradigm. Both languages follow the ECMAScript class syntax, but TypeScript enhances it with strong typing.

**Constructor Overloading (TypeScript) ?**

JavaScript does not support constructor overloading, but TypeScript allows it using method overloading.

**Understanding the Constructor**

* The constructor is a special method in a class that is automatically called when a new object is created using new ClassName().
* It initializes object properties (this.name and this.age).
* Unlike other methods, constructor does not return anything explicitly.

**Key Features of JavaScript Classes**

* Encapsulation: Class properties and methods are encapsulated within the class.
* Inheritance: One class can inherit from another.
* Static Methods: Methods that belong to the class, not instances.

**Access Modifiers in TypeScript**

|  |  |
| --- | --- |
| Modifier | Description |
| public | Accessible everywhere (default). |
| private | Accessible only within the class. |
| protected | Accessible within the class and subclasses. |

**Abstract Classes (TypeScript)**

Abstract classes provide a base structure but cannot be instantiated directly.

**Static Methods and Properties**

Static methods belong to the **class itself** rather than instances.

|  |  |  |
| --- | --- | --- |
| **Feature** | **JavaScript** | **TypeScript** |
| Class Syntax | ✅ | ✅ |
| Constructor | ✅ | ✅ (with type safety) |
| Access Modifiers | ❌ | ✅ (public, private, protected) |
| Readonly Properties | ❌ | ✅ (readonly) |
| Constructor Overloading | ❌ | ✅ (via method signatures) |
| Static Methods & Propertie | ✅ | ✅ |
| Inheritance | ✅ | ✅ |
| Abstract Classes | ❌ | ✅ |

**Conclusion**

* **JavaScript** provides basic class and constructor features.
* **TypeScript** enhances JavaScript classes with **strong typing, access modifiers, method overloading, and abstract classes**, making it more suitable for large-scale applications.

**ReadOnly property** – You can not change .

**?** – It is used for optional parameter in  constructor