

TypeScript

What is TypeScript?

- TypeScript is a **superset of JavaScript**, which means:
 - It contains **everything JavaScript offers**, plus additional features.
 - It's developed and maintained by **Microsoft**.
 - It contains the type system
- TypeScript makes writing JavaScript **easier, more structured, and less error-prone**.

Why Use TypeScript?

1. Additional Features Over JavaScript

- Supports advanced concepts like:
 - **Interfaces**
 - **Access Modifiers** (public, private, protected)
 - **Decorators**
 - **Enums**
- These features are **not available in standard JavaScript** (as of the latest ECMAScript versions).
- Makes complex applications easier to structure and scale.

2. Strong Typing System

- TypeScript allows developers to **define types** for variables and functions.
- **Strongly typed**: once a variable's type is set, it **cannot be changed**.
- Prevents type-related bugs before runtime.

Example:

```
let num: number = 10;  
num = "hello"; // ❌ Error: Type 'string' is not assignable to type  
                'number'
```

3. Early Error Detection

- Errors are caught during **compile-time**, not at runtime.
- Reduces unexpected crashes and bugs in production.
- Encourages better coding practices.

4. IDE & Developer Tooling Support

- Provides **intelligent code completion**, **type hints**, and **documentation** inside your editor.
- Works seamlessly with **Visual Studio Code**, **WebStorm**, and other IDEs.

5. Improved Readability & Maintainability

- Easier to understand what data types are expected and returned.
- Simplifies **refactoring** and **debugging** in large codebases.

6. Highly Configurable Compiler

- TypeScript uses the `tsconfig.json` file for configuration.
- You can set:
 - Output ECMAScript version (`ES5`, `ES6`, etc.)
 - Module system (`CommonJS`, `ESModule`)
 - Strictness flags (`strict`, `noImplicitAny`, etc.)
 - File paths, includes/excludes

How Does TypeScript Work?

1. **Write** code in TypeScript using advanced features.
2. **Compile** using the TypeScript compiler (`tsc`).
3. **Output** is JavaScript that can run in any JS environment (browsers, Node.js).

TypeScript Compiler converts new syntax/features into JavaScript using:

- Existing JavaScript features
- Workarounds and transformations

Example – Compiled Code Comparison

TypeScript:

```
class Person {  
  private name: string;  
  constructor(name: string) {  
    this.name = name;  
  }  
}
```

JavaScript (compiled):

```
"use strict";  
class Person {  
  constructor(name) {  
    this.name = name;  
  }  
}
```

Note: `private` keyword disappears, as JS doesn't support it directly — TS uses it for development-time checks.



TypeScript vs JavaScript – Typing Comparison

Feature	JavaScript	TypeScript
Typing	Dynamic (inferred at runtime)	Static (set during development)
Type Safety	No	Yes
Error Detection	At runtime	At compile-time
Refactoring Support	Minimal	Strong

Example – Dynamic Typing in JavaScript

```
let val = 100;    // val is number
val = "hello";    // val becomes string -- allowed in JS
```

Example – Static Typing in TypeScript

```
let val: number = 100;
val = "hello";    // ❌ Error: Type mismatch
```

Advantages of Using TypeScript

Type Safety & Early Bug Detection

- Reduce runtime issues by validating code before execution.

Advanced Features

- Interfaces, modifiers, decorators, generics, enums, etc.

Better Tooling

- Rich IntelliSense, code navigation, and error hints.

Compiler Configurability

- Customize the behavior of the TypeScript compiler for your project.

Disadvantage of TypeScript

- **TypeScript cannot run directly** in browsers or Node.js.
- Requires an **extra compilation step** using the TypeScript compiler (**tsc**) to convert **.ts** → **.js**.