**Introduction to TypeScript**

TypeScript is JavaScript’s runtime with a compile-time type checker.

TypeScript is a statically typed superset of JavaScript developed by Microsoft. It adds optional static typing to the language, which helps developers catch errors early during the development process. TypeScript code is transpiled into plain JavaScript, making it compatible with all browsers and JavaScript engines.

**Key Features:**

* **Static Typing**: TypeScript allows you to define the types of variables and function parameters, helping to catch type-related errors at compile time.
* **Modern JavaScript Features**: TypeScript includes features from the latest JavaScript versions (ES6 and beyond) while maintaining compatibility with older versions.
* **Tooling**: Enhanced development experience with support for better IDE features like auto-completion, navigation, and refactoring.
* **Object-Oriented Programming**: TypeScript supports OOP concepts like classes, interfaces, and inheritance.

**Setting Up Development Environment**

To start using TypeScript, you need to set up your development environment:

**Step 1: Install Node.js and npm**

TypeScript requires Node.js and npm (Node Package Manager). Download and install them from [Node.js official website](https://nodejs.org/).

**Step 2: Install TypeScript**

Open your terminal or command prompt and run the following command to install TypeScript globally:

Command>npm install -g typescript

**Step 3: Verify Installation**

Check the installed TypeScript version:

Command>tsc -v

**Step 4: Set Up a TypeScript Project**

1. Create a new directory for your project:
2. Initialize a new Node.js project:
3. Create a tsconfig.json file to configure TypeScript compiler options:
4. Create source and output directories:
5. Write your TypeScript code in the src directory and compile it using:

**Data Types**

TypeScript supports various data types, similar to JavaScript but with additional static typing.

**Basic Types:**

* **Boolean**: true or false

let bool: boolean = false;

* **Number**: All numbers are floating-point values.

let decimal: number = 6;

let hex: number = 0xf00d;

let binary: number = 0b1010;

let octal: number = 0o744;

* **String**: Represents textual data.

let color: string = "blue";

* **Array**: Collection of values of the same type.

let list: number [] = [1, 2, 3];

let anotherList: Array<number> = [1, 2, 3];

let hello: string [] =[“hello”,”world”];

* **Tuple**: An array with a fixed number of elements of different types.

let x: [string, number];

x = ["hello", 10];

* **Enum**: A way of giving more friendly names to sets of numeric values.

enum Color {Red, Green, Blue}

let c: Color = Color.Green;

* **Any**: A type for values that can be of any type, useful when working with third-party libraries.

let notSure: any = 4;

* **Void**: Represents the absence of any type, commonly used as the return type of functions that do not return a value.

function warnUser(): void {

console.log("This is my warning message");

}

The above function return type is void.

* **Null and Undefined**: Represent null and undefined values.

let u: undefined = undefined;

let n: null = null;

* **Never**: Represents the type of values that never occur, used for functions that never return or always throw an exception.

function error1(message: string): never {

throw new Error(message);

}

**Variables**

Variables in TypeScript can be declared using three keywords: var, let, and const.

**var**

* **Scope**: var declarations are function-scoped or globally-scoped, not block-scoped.
* **Hoisting**: var declarations are hoisted to the top of their containing scope.

**let**

* **Scope**: let declarations are block-scoped.
* **Hoisting**: let declarations are hoisted but not initialized, meaning they cannot be used before they are declared.

**const**

* **Scope**: const declarations are block-scoped.
* **Hoisting**: const declarations are hoisted but not initialized.
* **Immutability**: Variables declared with const cannot be reassigned.

const pi = 3.14;

// pi = 3.1415; // Error: Assignment to constant variable.

While const variables cannot be reassigned, they are not immutable. If the const variable is an object or array, its properties or elements can be modified.