What is TypeScript?

Definition:

TypeScript is an open-source programming language developed by Microsoft. It is a **superset of JavaScript** that adds optional **static typing** and **object-oriented features**.
Explanation:

TypeScript code gets **compiled to JavaScript**, which runs in any browser or JavaScript environment. It helps in catching errors during development before running the code.
Example:

```
let message: string = "Hello, TypeScript!";
console.log(message);
```

Benefits of TypeScript

- **Static Typing**: Helps catch type-related bugs during development.
- **Enhanced IDE Support**: Better autocompletion and refactoring tools.
- **OOP Features**: Supports classes, interfaces, inheritance, etc.
- **ES6+ Support**: Lets you use modern JavaScript features with backward compatibility.
- **Better Code Organization**: Improves scalability for large projects.

Setup the Environment

1. **Install Node.js** from https://nodejs.org
2. **Install TypeScript Compiler**:

   npm install -g typescript

3. **Create a .ts file and compile**:

   tsc app.ts

Basic Data Types

Explanation:

TypeScript supports built-in data types like number, string, boolean, etc.

Example:

```
let id: number = 1;
let name: string = "Alice";
let isAdmin: boolean = true;
```

Arrays

Explanation:

Arrays store multiple values in a single variable. Can be defined using two syntaxes.

Example:

```
let scores: number[] = [95, 85, 76];
let fruits: Array<string> = ["Apple", "Banana"];
```

Tuples

Explanation:

Tuples allow you to express an array with fixed number of elements of known types.

Example:

```
let user: [number, string] = [1, "Alice"];
```

Enum

Explanation:

Enums are used to define named constants.

Example:

```
enum Direction {
  Up,
  Down,
  Left,
  Right
}
let move: Direction = Direction.Left;
```

Any and void

Explanation:

- any allows a variable to hold any type (unsafe but flexible).

- `void` represents the absence of a value, mainly used for functions.

Example:

```
let value: any = 5;
value = "a string";

function greet(): void {
  console.log("Hello");
}
```

---

null and undefined

Explanation:

These are special types representing "no value".

- `null` – intentional absence of a value.
- `undefined` – variable declared but not assigned.

Example:

```
let name: string | null = null;
let age: number | undefined;
```

Type Inference

Explanation:

TypeScript can automatically infer a variable's type based on its initial value.

Example:

```
let country = "India"; // Inferred as string
```

Type Casting

Explanation:

Used to override the inferred type of a variable.

Example:

```
let code: any = 123;
let employeeCode = <number>code;
```

Difference between let and var

Explanation:

- let is block-scoped.
- var is function-scoped and hoisted.

Example:

```
function example() {
  if (true) {
    let x = 10;
    var y = 20;
  }
  // console.log(x); // Error
  console.log(y);    // Works
}
```

Const Declaration

Explanation:

const creates a read-only reference to a value. It must be initialized during declaration.

Example:

```
const PI = 3.14;
// PI = 3.141; // Error
```

Writing and Using Classes

Explanation:

A class defines a blueprint for creating objects.

Example:

```
class Student {
  name: string;

  constructor(name: string) {
    this.name = name;
  }

  display() {
    console.log(`Name: ${this.name}`);
  }
}
```

Constructor Method

Explanation:

The constructor method is automatically called when an object is created from a class.

Example:

```
let s1 = new Student("Bob");
s1.display();
```

🧬 Inheritance of Classes

Explanation:

A class can inherit properties and methods from another class using extends.

Example:

```
class Animal {
  move() {
    console.log("Moving...");
  }
}

class Dog extends Animal {
  bark() {
    console.log("Bark!");
  }
}

let pet = new Dog();
pet.move();
pet.bark();
```

Type Casting (again for review)

Example:

```
let input: any = "100";
let length: number = (input as string).length;
```

Type Assertion

Explanation:

Used to tell the compiler the specific type of a variable.

Example:

```
let value: any = "hello";
let length = (value as string).length;
```

Abstract Class

Explanation:

An abstract class cannot be instantiated and must be extended. It may contain abstract methods without implementation.

Example:

```
abstract class Shape {
  abstract area(): number;
}

class Circle extends Shape {
  constructor(public radius: number) {
    super();
  }

  area(): number {
    return Math.PI * this.radius ** 2;
  }
}
```

Interface Declaration and Initialization

Explanation:

An interface defines a contract for the structure of an object or class.

Example:

```
interface Employee {
  id: number;
  name: string;
  department?: string; // optional
}

let emp: Employee = {
  id: 101,
  name: "John"
};
```