Table of Contents

1. Installation and Setup

Install TypeScript (Globally):

```
npm install -g typescript
```

Check Version:

```
tsc -v
```

Initialize TypeScript Project (Creates tsconfig.json):

```
tsc --init
```

---

2. Compile TypeScript to JavaScript

```
tsc index.ts
```

- Compiles index.ts to index.js.
- Watch Mode (Auto-compile on save):

```
tsc --watch
```

---

3. Basic TypeScript Example

index.ts

```
function greet(name: string): string {
    return `Hello, ${name}`;
}

console.log(greet("Alice"));
```

Compile and Run:

```
tsc index.ts
```

```
node index.js
```

---

4. Types in TypeScript

1. Primitive Types:

```
let isDone: boolean = true;
let age: number = 25;
let firstName: string = "John";
let u: undefined = undefined;
let n: null = null;
```

2. Arrays:

```
let numbers: number[] = [1, 2, 3];
let fruits: Array<string> = ["Apple", "Banana"];
```

3. Tuples (Fixed-Length Array):

```
let person: [string, number] = ["Alice", 30];
```

4. Enums:

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

5. Any (Avoid if Possible):

```
let randomValue: any = 10;
randomValue = "String";  // No Error
```

---

5. Functions in TypeScript

1. Basic Function:

```
function add(a: number, b: number): number {
    return a + b;
}
```

2. Optional Parameters:

```
function multiply(a: number, b?: number): number {
    return b ? a * b : a;
}
```

3. Default Parameters:

```
function greet(name: string = "Guest"): string {
    return `Hello, ${name}`;
}
```

4. Rest Parameters:

```
function sum(...numbers: number[]): number {
    return numbers.reduce((acc, n) => acc + n, 0);
}
```

---

6. Interfaces

```
interface User {
    id: number;
    name: string;
    email?: string;  // Optional Property
}

const user: User = {
    id: 1,
    name: "John Doe"
```

```
};
```

---

7. Classes

```
class Person {
    name: string;
    age: number;

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

    greet() {
        console.log(`Hello, my name is ${this.name}`);
    }
}

const person1 = new Person("Alice", 28);
person1.greet();
```

---

8. Access Modifiers

```
class Employee {
    public name: string;
    private salary: number;
    protected position: string;

    constructor(name: string, salary: number, position: string) {
        this.name = name;
        this.salary = salary;
        this.position = position;
    }
}
```

- public – Accessible from anywhere.
- private – Accessible only within the class.
- protected – Accessible within the class and subclasses.

---

9. Inheritance and Extending Classes

```
class Animal {
    makeSound() {
        console.log("Animal sound");
    }
}

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

const dog = new Dog();
dog.makeSound();  // Output: Bark
```

---

10. Type Assertions (Type Casting)

```
let value: any = "Hello World";
let strLength: number = (value as string).length;
```

---

11. Generics

```
function identity<T>(arg: T): T {
    return arg;
}
```

```
let output1 = identity<string>("Hello");
let output2 = identity<number>(42);
```

---

12. Utility Types

```
interface User {
    id: number;
    name: string;
    email: string;
}

// Partial – Makes all properties optional
let userUpdate: Partial<User> = { name: "Bob" };

// Readonly – Prevents modification
const user1: Readonly<User> = { id: 1, name: "Alice", email:
"alice@email.com" };
```

---

13. Union and Intersection Types

1. Union Type:

```
let id: number | string;
id = 101;
id = "A101";
```

2. Intersection Type:

```
interface ErrorHandling {
    success: boolean;
    error?: { message: string };
}

interface Data {
```

```
    data: string[];
}

type ApiResponse = ErrorHandling & Data;
```

---

14. Type Guards

```
function isNumber(x: any): x is number {
    return typeof x === "number";
}

function printValue(value: number | string) {
    if (isNumber(value)) {
        console.log(value.toFixed(2));  // Number
    } else {
        console.log(value.toUpperCase());  // String
    }
}
```

---

15. Async/Await and Promises

```
async function fetchData(): Promise<string> {
    return new Promise((resolve) => {
        setTimeout(() => resolve("Data loaded"), 2000);
    });
}

fetchData().then((data) => console.log(data));
```

---

16. Working with Modules

Export Module:

```
export function add(a: number, b: number): number {
    return a + b;
}
```

Import Module:

```
import { add } from './math';
console.log(add(5, 10));
```

---

17. Common TypeScript Commands

```
tsc --init                      # Initialize TypeScript Project
tsc                             # Compile all .ts files
tsc app.ts                      # Compile a single file
tsc --watch                     # Watch for changes
```

---

18. tsconfig.json (Key Settings)

```
{
  "compilerOptions": {
    "target": "es6",
    "module": "commonjs",
    "strict": true,
    "outDir": "./dist",
    "esModuleInterop": true
  }
}
```