Introduction to TypeScript

An introduction to TypeScript

Yoosuf, 25-03-2025

What is TypeScript?

- TypeScript is a superset of JavaScript that adds static types.
- Helps catch errors early and improves code maintainability.
- Compiles to plain JavaScript, running anywhere JS run

Why Use TypeScript?

- Type safety
- Better IDE support (autocompletion, refactoring, debugging)
- Scalable and maintainable code
- OOP features like interfaces and classes

Setting Up TypeScript

1. To install TypeScript globally, run:

```
$ npm install -g typescript
```

2. Creating a TypeScript Project

```
$ mkdir my-typescript-project && cd my-typescript-project
```

3. Generate a tsconfig.json file:

```
$ npx tsc --init
```

Setting Up TypeScript

4. Create a src folder and add a main.ts file.

5. Compile TypeScript to JavaScript:

```
$ npx tsc
```

6. Run the compiled JavaScript file:

```
$ node dist/main.js
```

TypeScript Basics
~ Declaring Variables

```
let personName: string = "John";

let age: number = 25;

let isActive: boolean = true;
```

TypeScript Basics
~ Functions with Types

```
function greet(name: string): string {

    return `Hello, ${name}!`;

}

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

OOP in TypeScript

```
class Person {
    name: string;
    age: number;
    constructor(name: string, age: number) {
        this.name = name;
        this.age = age;
    }
    describe(): string {
        return `${this.name} is ${this.age} years old.`;
    }
}
const john = new Person("John Doe", 30);
console.log(john.describe());
```

OOP in TypeScript

```
const john = new Person("John Doe", 30);

console.log(john.describe());
```

Extending Person: Creating Customer

Customer Class

```
class Customer extends Person {

    customerId: number;

    constructor(name: string, age: number, customerId: number) {

        super(name, age);

        this.customerId = customerId;

    }

    describe(): string {

        return `${super.describe()} Customer ID: ${this.customerId}.`;

    }

}
```

Extending Person: Creating Administrator

Administrator Class

```
class Administrator extends Person {
    role: string;
    constructor(name: string, age: number, role: string) {
        super(name, age);
        this.role = role;
    }
    describe(): string {
        return `${super.describe()} Role: ${this.role}.`;
    }
}
```

Using the Classes

```
const alice = new Customer("Alice", 28, 1001);

const bob = new Administrator("Bob", 40, "Manager");


console.log(alice.describe());

console.log(bob.describe());
```

Interfaces in TypeScript

```typescript
interface Identifiable {

    id: number;

    getId(): number;

}
```

Interfaces in TypeScript
~ Implementing in Customer:

```
class PremiumCustomer extends Customer implements Identifiable {

    id: number;

    constructor(name: string, age: number, customerId: number, id: number) {

        super(name, age, customerId);

        this.id = id;

    }

    getId(): number {

        return this.id;

    }

}
```

Additional Resources to Learn TypeScript

1. Official TypeScript Documentation:
   https://www.typescriptlang.org/docs/

2. YouTube Video Tutorial: TypeScript Crash Course
   https://www.youtube.com/watch?v=BCg4U1FzODs

3. TypeScript Handbook:
   https://www.typescriptlang.org/handbook/

1. TypeScript enhances JavaScript with strong typing.
2. OOP principles like classes and interfaces make code reusable and structured.
3. Extending classes (Customer, Administrator) shows real-world applications.
4. TypeScript is the future of JavaScript development. Moving forward, OpusXanta will be using TypeScript as the standard for all projects.

Q & A