Getting Started with TypeScript

1. Installation

You can install TypeScript globally using npm:

```
npm install -g typescript
```

You can also create a new project and install TypeScript locally:

```
mkdir my-typescript-project
cd my-typescript-project
npm init -y
npm install --save-dev typescript
```

2. Setting Up a TypeScript Configuration File

Create a `tsconfig.json` file to configure your TypeScript project:

```
npx tsc --init
```

This will create a basic `tsconfig.json` file where you can customise compiler options.

3. Basic Types

TypeScript provides several basic types. Here are some of them:

Type	Description	Example
`string`	Textual data	`let name: string = "Alice";`
`number`	Numeric data	`let age: number = 30;`
`boolean`	True/false values	`let isActive: boolean = true;`
`any`	Any type	`let randomValue: any = 5;`
`void`	No return value	`function log(): void { console.log("Hello!"); }`
`array`	Array of items	`let numbers: number[] = [1, 2, 3];`
`tuple`	Fixed-size array	`let tuple: [string, number] = ["Alice", 30];`

4. Hands-On: Basic Types

Create a file named `basicTypes.ts` and add the following code:

```
let name: string = "Alice";
let age: number = 30;
let isActive: boolean = true;

console.log(`Name: ${name}, Age: ${age}, Active: ${isActive}`);
```

Compile the TypeScript code to JavaScript:

```
npx tsc basicTypes.ts
```

Run the output JavaScript file:

```
node basicTypes.js
```

5. Functions and Type Annotations

TypeScript allows you to define types for function parameters and return values:

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

console.log(add(5, 10));
```

6. Hands-On: Functions

Create a file named `functions.ts`:

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

console.log(multiply(5, 4));
```

Compile and run it the same way as before.

7. Interfaces

Interfaces allow you to define the shape of an object. Here's an example:

```
interface Person {
    name: string;
    age: number;
}

const person: Person = {
    name: "Alice",
    age: 30
};

console.log(person);
```

8. Hands-On: Interfaces

Create a file named `interfaces.ts`:

```
interface Car {
    make: string;
    model: string;
    year: number;
}

const myCar: Car = {
    make: "Toyota",
    model: "Corolla",
    year: 2020
};

console.log(myCar);
```

9. Classes

TypeScript supports ES6 classes with additional features like access modifiers:

```
class Animal {
    private name: string;

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

    public speak(): void {
        console.log(`${this.name} makes a noise.`);
    }
}

const dog = new Animal("Dog");
dog.speak();
```

10. Hands-On: Classes

Create a file named `classes.ts`:

```
class User {
    private username: string;

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

    public greet(): void {
        console.log(`Hello, ${this.username}!`);
    }
}

const user = new User("Alice");
user.greet();
```

11. Generics

Generics allow you to create reusable components. Here's a simple example:

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

```
}

console.log(identity<string>("Hello TypeScript"));
```

12. Hands-On: Generics

Create a file named `generics.ts`:

```
function wrapInArray<T>(value: T): T[] {
    return [value];
}

console.log(wrapInArray<number>(5));
console.log(wrapInArray<string>("Hello"));
```

Conclusion

By following these steps, you've covered the basics of TypeScript! Here's a quick summary of what you've learned:

- **Basic types and type annotations**
- **Functions and their types**
- **Interfaces for type safety**
- **Classes and access modifiers**
- **Generics for reusable components**