Các kiểu dữ liệu trong TypeScript

TypeScript cung cấp nhiều kiểu dữ liệu, bao gồm các kiểu cơ bản, các kiểu nâng cao và các kiểu do người dùng định nghĩa.

Kiểu cơ bản

string: Chuỗi

```
let name: string = "TypeScript"
```

number: Số nguyên, số thập phân

```
let age: number = 30;
```

boolean: Giá trị logic

```
let isPublished: boolean = true;
```

null: Giá trị null

```
let empty: null = null;
```

undefined: Giá trị chưa xác định

```
let notDefined: undefined = undefined;
```

symbol: Giá trị duy nhất và không thể thay đổi

```
let sym: symbol = Symbol("unique");
```

Kiểu phức tạp

any

Kiểu dữ liệu có thể là bất kỳ kiểu nào. Tránh sử dụng nếu có thể vì sẽ mất đi tính năng kiểm tra kiểu của TypeScript

```
let anything: any = "Could be anything";
```

unknown

Tương tự `any`, nhưng an toàn hơn vì yêu cầu kiểm tra kiểu trước khi sử dụng

```
let notSure: unknown = 4;
if (typeof notSure === "number") {
  let num: number = notSure; // Được phép vì đã kiểm tra kiểu
}
```

void

Thường dùng cho các hàm không trả về giá trị

```
function logMessage(): void {
  console.log("This is a message");
}
```

never

Đại diện cho kiểu không bao giờ xảy ra, thường dùng cho các hàm không bao giờ trả về (như các hàm ném ra ngoại lệ)

```
function error(message: string): never {
  throw new Error(message);
}
```

Kiểu cấu trúc

array: Mảng dữ liệu

```
let numbers: number[] = [1, 2, 3];
```

tuple: Mảng có độ dài cố định với các kiểu phần tử cụ thể

```
let tuple: [string, number] = ["hello", 10];
```

object: Đối tượng

```
let person: { name: string; age: number } = {
  name: "John",
  age: 25
};
```

Kiểu do người dùng định nghĩa

enum

Tập hợp các giá trị hằng số có tên

```
enum Color {
  Red,
  Green,
  Blue
```

```
}
let color: Color = Color.Green;
```

type

Cho phép định nghĩa kiểu mới từ các kiểu dữ liệu có sẵn

```
type Point = {
  x: number;
  y: number;
};
let point: Point = { x: 10, y: 20 };
```

interface

Tương tự như `type`, nhưng thường dùng để định nghĩa các cấu trúc đối tượng

```
interface Person {
  name: string;
  age: number;
}
let person: Person = { name: "Alice", age: 30 };
```

Kiểu nâng cao

union

Biểu diễn một biến có thể có nhiều kiểu dữ liệu

intersection

Kết hợp nhiều kiểu thành một

```
interface A {
  a: string;
}
```

```
interface B {
  b: number;
}
let ab: A & B = { a: "hello", b: 42 };
```