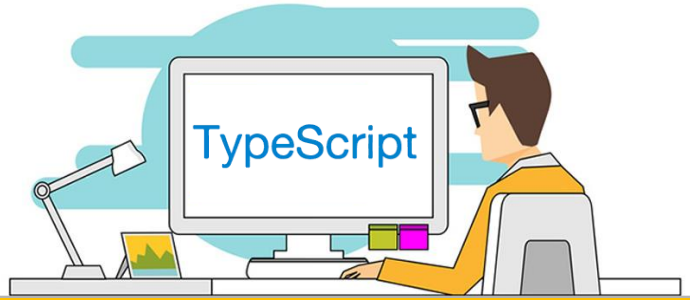
Buổi 4. TypeScript cơ bản

Giảng viên: Lê Đức Trung



NỘI DUNG CHÍNH

- Type system overview.
- Type alias vs interface.
- Function
- Enum
- Generics

MỤC TIÊU

- Tìm hiểu về các kiểu dữ liệu thường gặp trong TS.
- Type Alias và Interface.
- Function trong TS.
- Enum trong TS.
- Giới thiệu về generics

Type system overview (TS)

1. **Các kiểu dữ liệu bạn đã biết trong JS:**

- **Primitive**: number, boolean, string, null, undefined, symbol.
- **Reference**: array, object, function. any, unknown, void, never, ….

```
let number = 5;
number = 'string'
=> type error
```

Câu hỏi thảo luận : let string = 'string' ; string = 5;

2. **Cách khai báo biến trong TS :**

```
let count: number = 5;
let isActive: any = true;
```

Câu hỏi thảo luận : Có thể dùng giá trị làm kiểu dữ liệu trong TS (Literal types).
Vd: let count: 5 , phân biệt const count = 5 và let count = 5;

AIELA
NEXT INNOVATION

Type Alias vs Interface

1. Type Alias vs Interface

- Type Alias: là cách mà bạn định nghĩa ra type (kiểu).

```
type Person = {
  readonly id: number;
  name: string;
  isStatus?: boolean;
}
```

- Interface: là cách mà bạn định nghĩa ra object type.

```
interface Person {
  readonly id: number;
  name: string;
  isStatus?: boolean;
}
```

Câu hỏi thảo luận : Tìm sự khác nhau giữa type và interface?

AIELA
NEXT INNOVATION

Type Alias vs Interface

2. Union type: kết hợp 2 hoặc nhiều kiểu dữ liệu để tạo ra một kiểu dữ liệu mới.

```
type Status = 'active' | 'inactive'
type NumberString = 'number' | 'string'

let x: NumberString = 'test'; => work
let x: NumberString = 1; => work
let x: NumberString = true; => type error
```

Function trong TS

1. **Void return**

```
function sayHello(): void {
  console.log('Hello')
}
```

2. **Optional and default parameter**

```
function getLength(arrNumber?: number[]): number {
  return arrNumber ? arrNumber.length : +0;
}

function getLength(arrNumber: number[] = []): number {
  return arrNumber.length;
}
```

Lưu ý: Không thể kết hợp vừa optional và default parameter.

Enum trong TS

1. **Enum: tập hợp các giá trị cùng nhóm, dễ quản lý và truy xuất**

```
enum Roles {
   'Admin' = 1,
   'Mod' = 2,
   'User' = 3
}
```

Lưu ý: Enum dùng cơ chế number enum nếu như bạn khai báo dạng

```
enum Roles {
   'Admin', // 0
   'Mod',   // 1
   'User'   // 2
}
```

Lưu ý: Với number enum bạn có thể từ giá trị => khóa. Đối với string enum thì không.

Enum trong TS

1. **Enum: tập hợp các giá trị cùng nhóm, dễ quản lý và truy xuất**

```
enum Roles {
   'Admin' = 1,
   'Mod' = 2,
   'User' = 3
}
```

Lưu ý: Enum dùng cơ chế number enum nếu như bạn khai báo dạng

```
enum Roles {
   'Admin', // 0
   'Mod',   // 1
   'User'   // 2
}
```

Lưu ý: Với number enum bạn có thể từ giá trị => khóa. Đối với string enum thì không.

Generics

- **Generics: thêm giá trị vào cho kiểu dữ liệu.**

```
interface Array<T> {
  length: number;
  [index: number]: T;
}
```

Kết thúc

THANK FOR WATCHING

AIELA
NEXT INNOVATION