

## 5.1. Khái niệm

- **Typescript** là một ngôn ngữ lập trình mã nguồn mở, được xây dựng dựa trên JavaScript.
- **Typescript bổ sung** thêm các **kiểu dữ liệu** khi khai báo biến.



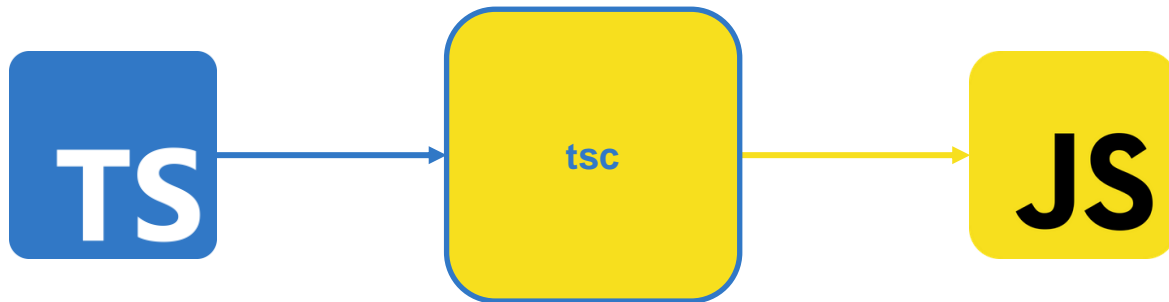
## 5.2. Cài đặt

cài typescript ở chế độ cục bộ

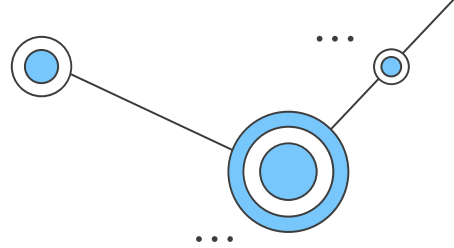
nếu không thể cài cục bộ thì cài cục bộ luôn

- Cài đặt TS: **npm i typescript**
- Biên dịch TS thành JS: **tsc ten-file.ts**

(npm i -g typescript)



## 5.3. Cấu hình tsconfig.json

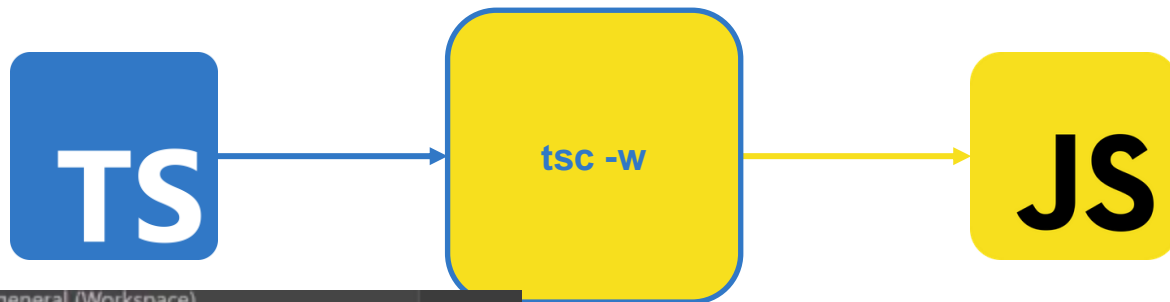


- Tạo file: **tsconfig.json**

- Biên dịch TS thành JS: **tsc -w**

biên dịch được nhiều file ts thành js

-w là viết tắt của từ watch nó xem xem có 1 file thay đổi thì nó tự biên dịch lại thành js

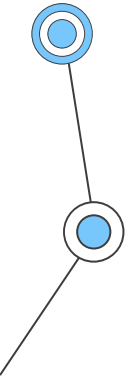
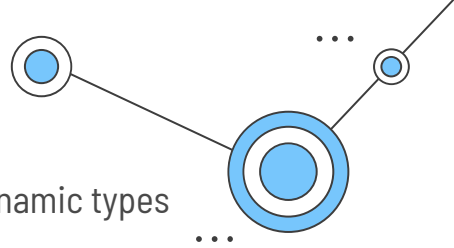


```
1 {  
2   "compilerOptions": {  
3     "outDir": "js",  
4   }  
5 }
```

biên dịch ts vào thư mục js

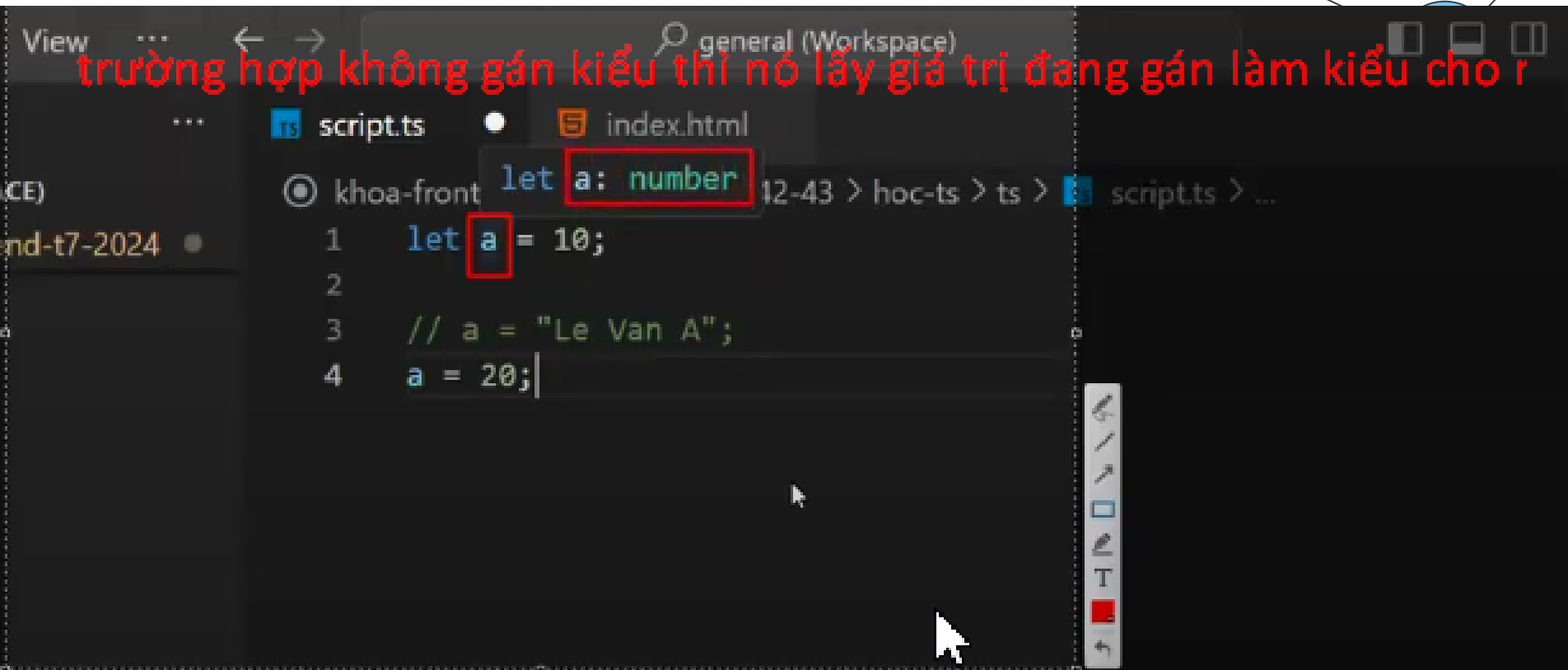
## 06. Tại sao sử dụng TypeScript?

- Typescript thêm các **types** khi khai báo biến để giúp tránh được nhiều vấn đề với dynamic types trong Javascript.
- *Hướng dẫn trong lúc học.*



## 7.1. Kiểu Number

trường hợp không gán kiểu thì nó lấy giá trị đang gán làm kiểu cho r



The screenshot shows the VS Code editor interface. The top bar includes the 'View' menu and a search icon. The file explorer on the left shows a project named 'khoa-front' with a file 'script.ts' selected. The editor displays the following TypeScript code:

```
1 let a = 10;  
2  
3 // a = "Le Van A";  
4 a = 20;
```

Annotations in the image include:

- A red box around the variable `a` in the first line (`let a = 10;`).
- A red box around the type `number` in the declaration `let a: number`.

The right sidebar shows the 'general (Workspace)' settings panel.

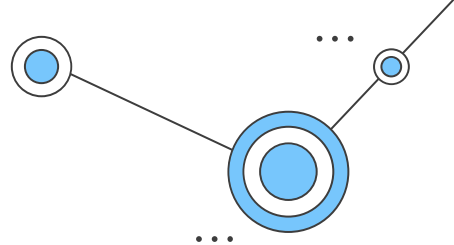
## 7.2. Kiểu String

```
0
1 // String
2 const s:string = "xin chào";
3 console.log(s);
4
```

## 7.3. Kiểu Boolean

```
// Boolean  
const b:boolean = false;  
console.log(b);
```

## 08. Kiểu Object



```
18
19 // object
20 const obj : {
21     // define type variable
22     fullName: string,
23     email: string,
24     tel: string
25 } = {
26     fullName: "Ke Lu Hanh",
27     email: "keluhanh@gmail.com",
28     tel: "123"
29 };
30
31 console.log(obj);
32
```



# 01. Interface

Vai trò như là một bộ khung

```
// interface
interface User {
  fullName: string,
  email: string,
  age: number,
  tel: string
};

const user1 : User = {
  fullName: "Ke lu hanh 1",
  email : "keluhanh@gmail.com",
  tel: "khong co",
  age: 15
};

console.log(user1);
```

// Bài 43

// 01. Interface

```
interface IUser {
  fullName: string;
  email: string;
  age?: number;
}
```

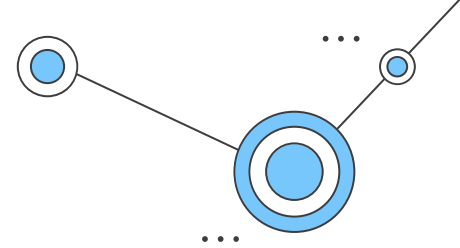
có dấu ? này là  
trường này được  
nhấn để trống

```
const userA: IUser = {
  fullName: "Le Van A",
  email: "levana@gmail.com",
  age: 20
};
```

```
const userB: IUser = {
  fullName: "Le Van B",
  email: "levanb@gmail.com"
};
```

```
console.log(userA);
console.log(userB);
```

## 02. Extend interface

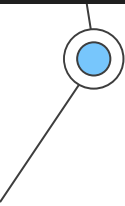


```
//Extend interface (kế thừa interface)
```

```
interface ITF1 {  
    id: string,  
    name: string,  
}
```

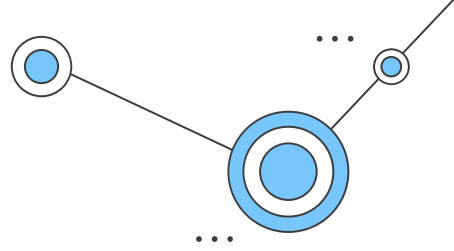
```
interface ROLE1 extends ITF1 {  
    role_id: string,  
    permission: string,  
}
```

```
const userRole : ROLE1 = {  
    id: "123",  
    name: "Nhan",  
    role_id: "1",  
    permission: "khong"  
}
```



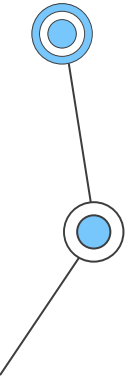
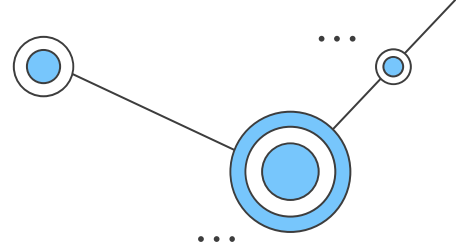
## 03. Kiểu Array

- Hướng dẫn trong lúc học.



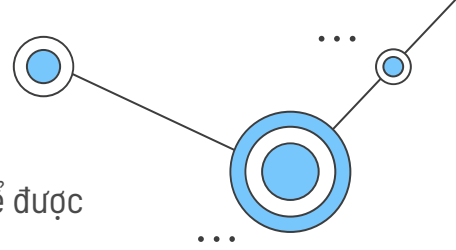
## 04. Kiểu Tuple

- Tuple giống như một mảng, nhưng:
  - Số lượng phần tử trong tuple là cố định.
  - **Types** của các phần tử trong mảng được chỉ định trước và không cần giống nhau.
- *Hướng dẫn trong lúc học.*



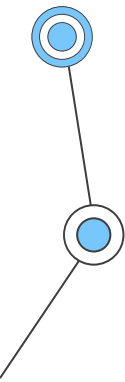
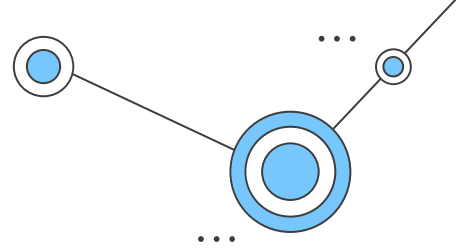
## 05. Readonly

- Từ khóa **readonly** được sử dụng để chỉ định rằng một biến hoặc thuộc tính không thể được gán lại sau khi nó đã được khởi tạo.
- *Hướng dẫn trong lúc học.*



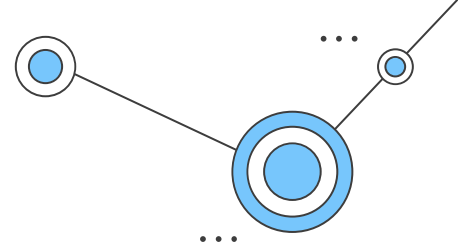
## 06. Functions

- *Hướng dẫn trong lúc học.*



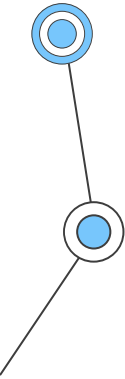
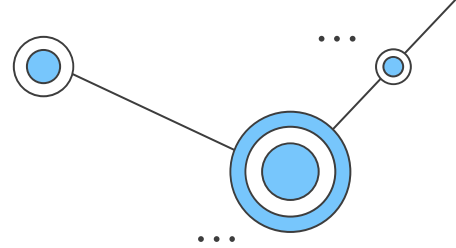
## 07. Default Parameters

- *Hướng dẫn trong lúc học.*



## 08. Rest Parameters

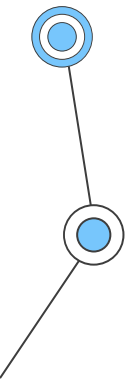
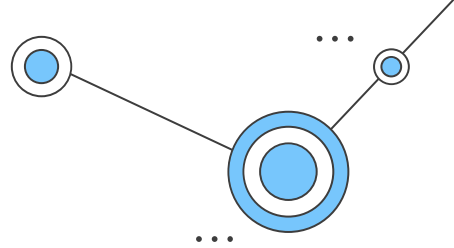
- *Hướng dẫn trong lúc học.*





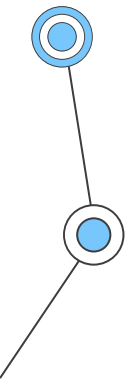
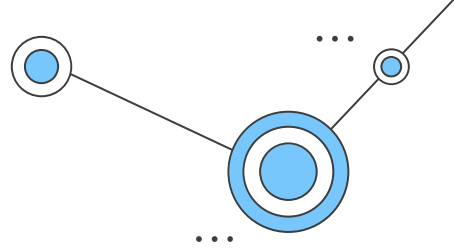
## 09. Kiểu Enum

- Kiểu **enum** để liệt kê một nhóm các giá trị constant.
- *Hướng dẫn trong lúc học.*



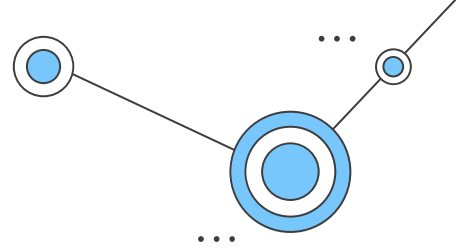
## 10. Kiểu Any

- Kiểu **any** giúp thay đổi linh hoạt được kiểu dữ liệu của một biến.
- *Hướng dẫn trong lúc học.*



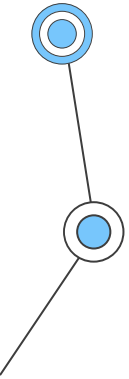
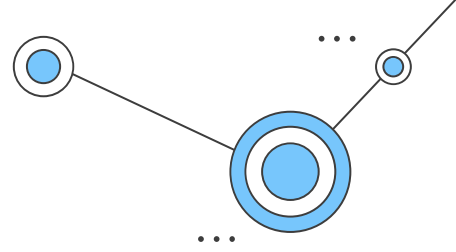
# 11. Kiểu Void

- Sử dụng kiểu **void** khi một hàm không return về giá trị.
- *Hướng dẫn trong lúc học.*



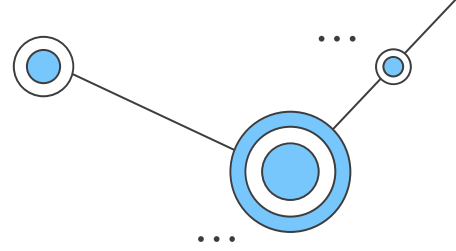
## 12. Union type

- Kết hợp nhiều kiểu dữ liệu để tạo thành 1 kiểu dữ liệu mới.
- *Hướng dẫn trong lúc học.*



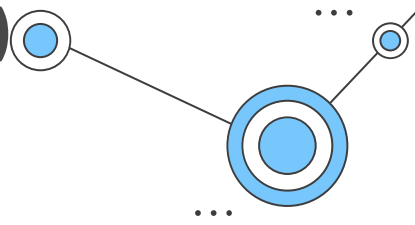
## 13. Type alias

- Type Aliases cho phép tạo ra một tên tùy chỉnh để đặt tên cho các kiểu dữ liệu.
- *Hướng dẫn trong lúc học.*



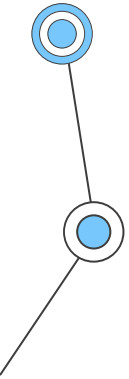
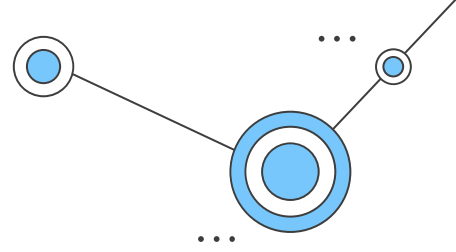
## 14. Intersection type (Hợp 2 Interface thành 1)

- Hướng dẫn trong lúc học.



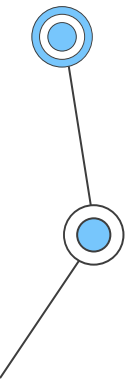
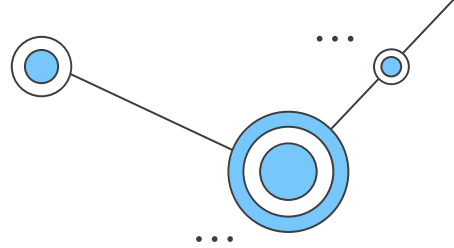
## 15. Declaration merging (Hợp 2 Interface trùng tên thành 1)

- *Hướng dẫn trong lúc học.*



## 16.1. Partial<Type>

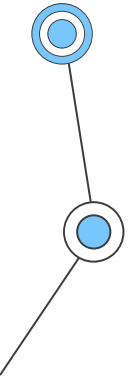
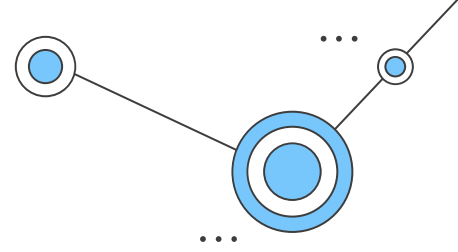
- Thay đổi tất cả các thuộc tính trong một đối tượng thành tùy chọn (optional).
- *Hướng dẫn trong lúc học.*





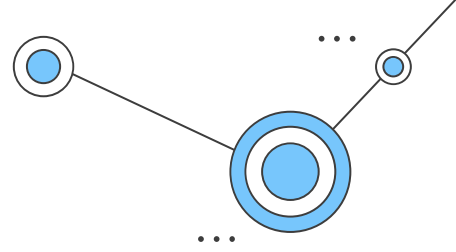
## 16.2. Required<Type>

- Thay đổi tất cả các thuộc tính trong một đối tượng thành bắt buộc.
- *Hướng dẫn trong lúc học.*



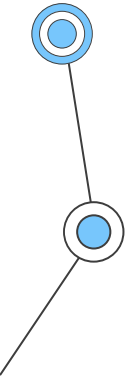
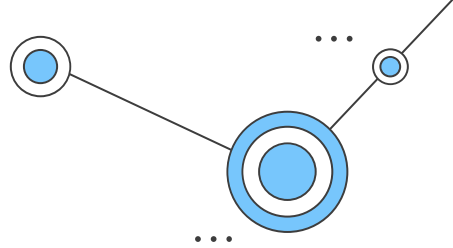
## 16.3. Omit<Type, Keys>

- Xóa một hoặc nhiều thuộc tính ra khỏi đối tượng.
- *Hướng dẫn trong lúc học.*



## 16.4. Pick<Type, Keys>

- Xóa tất cả các thuộc tính ra khỏi đối tượng, ngoại trừ các thuộc tính muốn giữ lại.
- *Hướng dẫn trong lúc học.*



## 16.5. Readonly<Type>

- Tất cả các thuộc tính trong đối tượng đổi thành trạng thái chỉ đọc, không sửa được.
- *Hướng dẫn trong lúc học.*

