# LOẠI CƠ BẢN

## Typescript là gì?

Là ngôn ngữ lập trình mở rộng của JavaScript, bổ xung các tính năng kiểu tĩnh và nhiều tính năng nâng cao giúp kiểm soát lỗi và phát triển ứng dụng dễ dàng hơn.

Lợi ích:

* Phát hiện lỗi ngay từ khi viết code thay vì đợi đến lúc chạy.
* Cải thiện đọc hiểu và bảo trì code.

## Kiểu chú thích

Chỉ định kiểu dữ liệu của biến, hàm, đối tượng.

Cú pháp:

: kiểu dữ liệu

VD1:

let counter: number;

counter = 1; // đúng

counter = "Hello"; // sai, biên dịch lỗi

VD2: Mảng.

let names: string[] = ['John', 'Jane', 'Peter', 'David', 'Mary'];

VD3: Đối tượng.

let person: {

name: string;

age: number;

};

person = {

name: 'John',

age: 25,

}; // valid

VD4: Hàm.

const sayHello = (name: string): string => {

return `Hello, ${name}!`

}

Có thể quy định 1 biến có 2 hoặc nhiều kiểu dữ liệu ngăn cách nhau bởi |

const userId = string | number.

## Kiểu suy luận

VD:

let counter = 0;

Tương đương với:

let counter: number = 0;

Đối với hàm nếu không có return sẽ mặc định quy định kiểu hàm là void, còn nếu có trả về thì nó sẽ dựa trên kiểu dữ liệu của return mà quy định kiểu trả về.

Đối với kiểu suy luận về cơ bản giống javascript, nó vẫn chạy được trong hầu hết các trường hợp, tuy nhiên đối với dự án lớn, đòi hỏi mức độ chính xác cao cần nên chỉ định kiểu.

## Kiểu số

Dùng kiểu : number cho tất cả các kiểu số, ngoài ra đối với số lớn hơn 253 – 1 thì dùng : bigint.

let price: number;

let big: bigint = 9007199254740991n;

## Kiểu chuỗi

Dùng kiểu : string, cho kiểu dữ liệu chuỗi.

let firstName: string = 'John';

let title: string = "Web Developer";

let profile: string = `I'm ${firstName}`;

## Kiểu đúng sai

Dùng kiểu : Boolean.

let completed: boolean = true;

## Kiểu đối tượng

Là kiểu đại diện cho tất cả giá trị không thuộc loại kiểu nguyên thủy.

Kiểu nguyên thủy: number, bigint, string, boolean, null, undefined, symbol

VD:

let employee: object;

employee = {

firstName: 'John',

lastName: 'Doe',

age: 25,

jobTitle: 'Web Developer'

};

## Kiểu mảng

Cú pháp:

let arrayName: type[];

VD:

let skills: string[] = [];

let scores : (string | number)[];

scores = ['Programming', 5, 'Software Design', 4];

## Kiểu Tuple

Hoạt động giống như mảng nhưng sẽ có thêm vài điều kiện.

VD:

let skill: [string, number];

skill = ['Programming', 5];

Nếu thay đổi thứ tự, chương trình sẽ lỗi.

Khai báo biến tùy chọn, đôi khi bạn không chắc chắn hoặc tham số đó có thể có hoặc không bạn có thể dùng ? để quy định nó:

let skill: [string, number?];

skill = ['Programming', 5]; // đúng

skill = ['Programming']; // đúng

// Tham số thứ 2 bắt buộc là kiểu number và có giá trị hoặc không đều không lỗi.

## Kiểu enum

Là đặt tên của một nhóm các giá trị hằng số (viết tắt của enumerated)

Cú pháp:

enum name {constant1, constant2, ...};

VD:

enum Month {

Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec,

}

console.log(Month.Jan);

// 0

enum Status {

Pending = 1, // Bắt đầu từ 1

InProgress = 3,

Completed = 5

}

console.log(Status.Pending); // 1

console.log(Status.Completed); // 5

Tuy nhiên enum thường không được sử dụng mấy trong dự án thực tế vì nó có khá nhiều nhược điểm:

* Một tập hợp nhỏ có các giá trị cố định
* Giá trị này không được thay đổi

Nên sử dụng type thay vì enum.

## Kiểu dữ liệu any

Là kiểu dữ liệu bất kỳ, typescript sẽ bỏ qua bắt lỗi đối với biến này, tức là biến này có thể gán bất kì giá trị nào, chỉ nên sử dụng trong trường hợp không biết trước hoặc chấp nhận mọi giá trị chấp nhận rủi ro

VD:

let variable: any;

let variable; // mặc định không gán kiểu hoặc giá trị sẽ là any

# KIỂM SOÁT LUỒNG TRẠNG THÁI

# HÀM

# LỚP

# INTERFACE

# LOẠI NÂNG CAO

# GENERIC

# MODULES

# CÀI ĐẶT CÔNG CỤ TYPESCRIPT