Báo cáo giai đoạn 1

Link Git : [Xem tại đây](https://github.com/tysdz/onthisinhvien-FE.git)

# **Tìm hiểu Typescript**

**TypeScript là Javscript**:

Bản chất của TypeScript là biên dịch tạo ra các đoạn mã javascript nên ban có thê chạy bất kì ở đâu miễn ở đó có hỗ trợ biên dịch Javascript. Ngoài ra bạn có thể sử dụng trộn lẫn cú pháp của Javascript vào bên trong TypeScript, điều này giúp các lập trình viên tiếp cận TypeScript dễ dàng hơn.

Có thể coi TypeScript là một phiên bản nâng cao của Javascript bởi việc bổ sung tùy chọn *kiểu tĩnh và lớp hướng đối tượng* mà điều này không có ở Javascript.

Điểm khác biệt lớn nhất của TypeScript so với Javascript là trong khi các biến của JavaScript là undefine trong khi các biến của TypeScript được khai báo ngay khi bắt đầu.

Trong TypeScript chia ra làm 7 loại cơ bản, bao gồm: boolean, number, string, array, enum, any, void. khi khai báo ta sẽ sử dụng cấu trúc như sau:

var tên\_biến : kiểu\_trả\_về = giá\_trị\_biến;

* Enum: khi khai báo enum một cách thông thường các phần tử sẽ được đánh số từ 0 và tăng dần
* Any: Any là một kiểu mà bạn có thể gán bất kỳ kiểu nào cho nó.

var list:any[] = [1, true, "free"];

* Void: Cũng giống như any nhưng void được sử dụng là đầu ra của hàm.

## Function

Ngoài cách khai báo hàm cơ bản như JavaScript, function typeScript còn hỗ chợ việc khai báo với các kiểu trả ra của function và cũng như kiểu đầu vào của dữ liệu:

Vd : function add(x: number, y: number): number {

return x+y;

}

var myAdd = function(x: number, y: number): number { return x+y; };

Bên cạnh đó là việc có thể khai báo giá trị đầu vào mặc định hoặc bỏ qua 1 vài giá trị đầu vào :

function buildName(firstName: string, lastName = "Smith") {

return firstName + " " + lastName;

}

And :

function buildName(firstName: string, lastName?: string) { (tương đương lastName với giá trị có thể là undefine)

if (lastName)

return firstName + " " + lastName;

else

return firstName;

}

* TypeScript còn có một cách làm việc với các param đầu vào có tên gọi: “Rest Parameters”
* Còn thêm về overloading function (sử dụng khi mà chúng ta có nhiều kiểu dữ liệu khác nhau nhưng kết quả cuối cùng trả về cùng 1 giá trị)

A computer screen with text

Description automatically generated

* Nếu chúng ta ko định nghĩa kiểu trả về của dữ liệu thì dữ liệu sẽ mặc định đc trả về String:

function add(x: number, y: number) {

return x+y;

} 🡺 kiểu dữ liệu là String

Điểm nổi bật : khi khai báo đc kiểu dữ liệu , nếu ta sử dụng hàm mà dùng sai kiểu dữ liệu, typescript sẽ báo lỗi cho chúng ta ngay khi ta khai báo, tiện lợi hơn rất nhiều so với javascript khi mà chạy xong thấy lỗi rồi ta mới đi tìm lỗi.

## Class

Hàm constructor sẽ được chạy ngay khi khởi tạo class mới. Hơn nữa khi ép khai báo class chúng ta cũng phải ép kiểu cho dữ liệu ngay vào lúc đó.

Các Access Modifiers : public, private, protected.

Khi sử dụng các Access Modifiers, sẽ báo code sai trước khi mà chúng ta triển khai.

private, protected ( cần khai báo còn public ko cần khai báo)

**Đặc biệt**, typescripts hỗ trợ Access Modifiers : ReadOnly (không thể UPDATE/DELETE)

Getter trong type là get

Setter trong type là set

# **Tìm hiểu về SCSS**

SCSS (Sassy CSS), một dạng mở rộng của CSS có cú pháp linh hoạt và tính năng mạnh mẽ hơn. SCSS giúp giảm bớt công việc lặp lại và làm cho mã nguồn CSS trở nên dễ đọc và dễ bảo trì hơn.

Những tiện lợi hơn của SCSS so với sử dụng CSS thuần :

## Import

Giả sử bạn có 1 trang index, bao gồm header, body, footer. Thay vì bạn CSS cho các phần trên vào 1 style.css thì với SASS bạn làm như sau:

* Tạo 1 file \_header.scss để CSS riêng cho header.
* Tạo 1 file \_body.scss để CSS riêng cho body.
* Tạo 1 file \_footer.scss để CSS riêng cho footer. (nhớ có dấu \_ trước tên file được import)

Rồi tất cả sẽ được import trong 1 file tổng style.scss

@import 'header';

@import 'body';

@import 'footer';

## Biến (Variable)

Biến chứa các giá trị mà bạn có thể sử dụng nhiều lần, ví dụ mã màu, font hay kiểu chữ....

Để khai báo một biến trong SASS, chúng ta sẽ viết dấu $ cùng với tên biến. Sau đó, khi muốn sử dụng ta chỉ cần gọi lại biến đó. Điều này rất tiện lợi giúp ta giảm bớt thời gian khi dùng lại các biến.

Vd: $whiteColor = #fff;

## Quy tắc xếp chồng (Nested Rule)

Trong CSS, khi mà chúng ta muốn CSS cho các phần tử con bên trong 1 lớp cha thì ta phải gọi các lớp cha đó ra trước khi muốn sử dụng lớp con. Cứ phải viết lặp đi lặp lại tên tag (hoặc class, hoặc id) cha của thẻ muốn CSS rất tốn thời gian và làm code trở lên nằng nhằng. Nested Rule của SASS sinh ra để giúp bạn làm điều trên một cách đơn giản hơn.

Tương tự với mô hình thác nước , Nested Rule được triển khai như sau :  
A screen shot of a computer code

Description automatically generated

Đặc biệt , xếp chồng cũng hỗ trợ chúng ta trong khi chúng ta làm responsive.

## Quy tắc Mixin

Có thể nói Mixin gần giống với quy tắc Variable ở trên, tuy nhiên, Mixin không chỉ lưu những dữ liệu cơ bản giống Variable. Công dụng của nó là mang nhiều thuộc tính mà bạn đã quy ước trong một mix nào đó rồi @include vào một thành phần bất kỳ mà không cần phải viết lại các thuộc tính đó ( ví dụ như cùng font-size, color, padding)

Các cách mà ta có thể dùng mixin :

* Truyền vào các giá trị cố định :

@mixin colorVsFont {

color: #fff;

font-size: 50px;

}

Rồi gọi ra ở phần nào mà chúng ta muốn áp dụng các thuộc tính trên:

a {

text-decoration: none;

@include colorVsFont;

}

* Truyền thuộc tính ( giống với các truyền params vào method )

@mixin colorVsFont($color, $fontSize) {

color: $color;

font-size: $fontSize;

}

Rồi gọi ra ở phần nào mà chúng ta muốn áp dụng các thuộc tính cùng với các giá trị ta muốn:

a {

text-decoration: none;

@include colorVsFont(#000, 50px);

}

## Extends – Kế thừa

Có thể nói đây là 1 loại biến thể của mixin khi mà ta có thể kế thừa từ các code đã viết ở phía trên mà không cần viết lại các thuộc tính trong css hay khai báo hẳn cả mixin bởi mixin là ta cần sử dụng nhiều lần trong khi Extends có thể sẽ cần dùng 1 đến 2 lần .

Vd: .title-box {

color: #dacb46;

text-shadow: 1px 1px 1px #1a1a1a;

display: inline-block;

text-transform: uppercase;

}

Thẻ a sẽ kế thừa title-box:

a {

text-decoration: none;

@extend .title-box;

}

Bởi vì SCSS thực hiện CSS từ trên xuống dưới nên ta có thể thay đổi một vài thuộc tính sau khi extend bằng cách ghi đè:

a {

text-decoration: none;

@extend .title-box; (color là màu #dacb46)

color : #fff ;

} Kết quả cuối cùng của color là #fff

## Hỗ trợ các biểu thức toán học

Một số biểu thức hỗ trợ bao gồm: +, -, /, \*, %, == và !=.

Lưu ý :

Nếu bạn muốn sử dụng variables cùng với một plain CSS /, bạn có thể sử dụng #{}.

Vd :

p {

$font-size: 12px;

$line-height: 30px;

font: #{$font-size}/#{$line-height};

}

Thứ hai, bạn không được kết hợp các đơn vị khác nhau trong trong một phép toán. (vd: 100% - 50px)

# **Áp dụng vào project**

## Áp dụng Typescipt

* Đã tạo trang giỏ hàng
* Đã thay đổi 90% code sang Typescript

Khó khăn :

Vẫn chưa thay đổi đc mã trong thư mục chính :

A screenshot of a computer

Description automatically generated

## Áp dụng SCSS

* Đã thay đổi mã code

Hạn chế:

* Mới chỉ áp dụng đc Nguyên lý chồng chất và 1 số kế thừa
* Chưa áp dụng được nhiều @mixin cũng như tạo module