**Tên Nhóm: NCP (Nhóm 1)**

Thành Viên:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Họ Và Tên | Email | SDT |
| 1 | Lê Đình Cường | [ledinhcuongse@gmail.com](mailto:ledinhcuongse@gmail.com) | 0359752007 |
| 2 | Đoàn Thị Yến Nhi | [Doanthiyennhi1001@gmail.com](mailto:Doanthiyennhi1001@gmail.com) | 0388102525 |
| 3 | Trần Đức Phương | [phuongtranitnl@gmail.com](mailto:phuongtranitnl@gmail.com) | 0937320311 |
| 4 | Bùi Đức Cơ | [Rongtrang137@gmail.com](mailto:Rongtrang137@gmail.com) | 01889361179 |

Email nhóm: ncpdidong3@gmail.com

Pass: Aa359752007

Github: <https://github.com/phuongtranitnl/didong3>

Đề tài: Xây dựng ứng dụng ghi chú

Tuần 1.2

Ngày 31/10/2018

---------------------------------------------------------------------------------

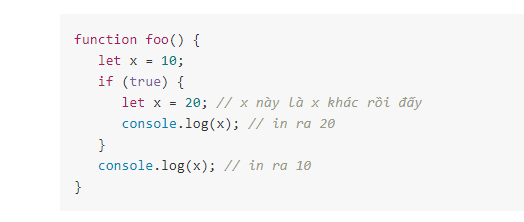
**Phần 1: Sự khác biệt giữa 3 loại biến let, var và conts**

**So sánh giữa biến Let và var và Const (Tác giả: Đoàn Thị Yến Nhi và Bùi Đức Cơ)**

**Biến Let**

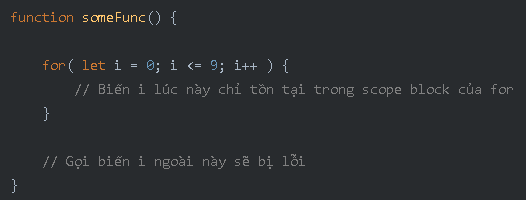
* Tạo ra một biến chỉ có thể truy cập được trong block bao quanh nó tức là trong phạm vi { }

VD:



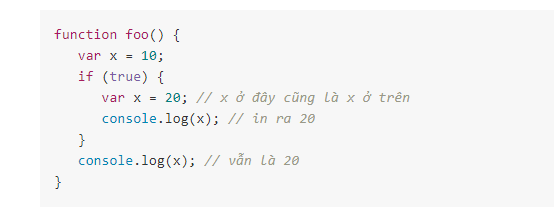
* Không cho phép khai báo lại biến cũ
* Tác động vào function block: chỉ tác động vào block ngay sau nó:

VD:



**Biến Var**

* Tạo ra một biến có phạm vi truy cập xuyên suốt function chứa nó.
  + VD:

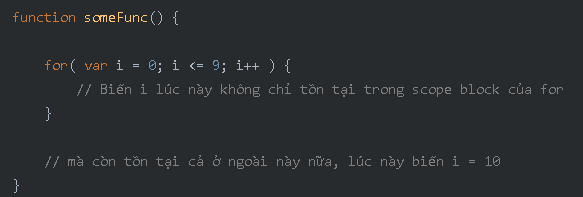


* Cho phép khai báo lại biến cũ:
* VD:

var n = 1 , var n = 2

- Tác động vào function block

* VD:

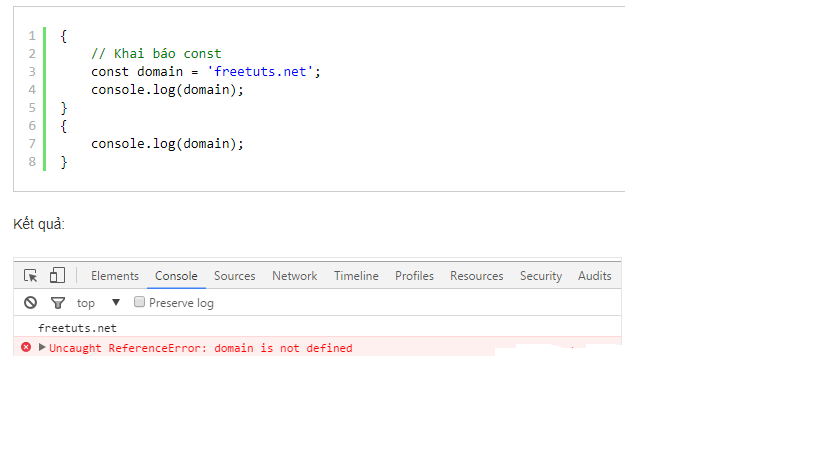


Khai báo hằng Const

* const là khai báo hằng số
* biến được khai báo là hằng số thì bạn phải gán giá trị lúc khai báo và không thể thay đổi giá trị cho biến đó được nữa

VD



* Chỉ tồn tại trong phạm vi nó được khai báo ,mà thôi
* vd:  
  

**Phần 2: Tìm hiều về object (Tác giả: Trần Đức Phương)**

# [ES6 Object Literal](https://www.codehub.vn/ES6-Co-Ban/ES6-Object-Literal)

## Object Literal Là Gì

Object trong JavaScript có thể được tạo theo một số cách khác nhau. Object literal là kiểu cú pháp tạo object sử dụng cặp dấu ngoặc {} và bên trong đó là danh sách các **property** (thuộc tính) của object.

Với phiên bản ECMAScript trước đây chúng ta thường sẽ

**var** name = "John";

**var** myObj = { name: name };

Các property có giá trị là một hàm còn được gọi là **method**:

**var** name = "John";

**var** myObj = {

name: name,

sayHi: **function** () {

console.log("Hi, my nam is " + **this**.name)

}

};

## Cú Pháp Object Literal Trong ES6

Trong ECMAScript 6 cú pháp khởi tạo object sử dụng object literal được đơn giản hoá hơn.

Với những property có tên trùng với giá trị như name:name ở trong ví dụ trên thì lập trình viên có thể thay thế bởi name:

**let** name = "John";

**let** myObj = { name };

Với các property nhận giá trị là một hàm (hay còn gọi là method) thì lập trình viên có thể thay thế bằng cách khai báo một với tên của property và đồng thời bỏ qua từ khoá function. Ví dụ với đoạn code sau khởi tạo object sử dụng kiểu cũ:

**var** name = "John";

**var** myObj = {

name: name,

sayHi: **function** () {

console.log("Hi, my nam is " + **this**.name)

}

};

myObj.sayHi()

*// Hi, my nam is John*

Thì trong ES6 có thể được viết lại như sau:

**let** name = "John";

**let** myObj = {

name,

sayHi() {

console.log("Hi, my nam is " + **this**.name)

}

};

myObj.sayHi()

*// Hi, my nam is John*

## Object () Constructor

JavaScript cung cấp một constructor function đặc biệt được gọi là Object() để xây dựng object. Toán tử mới được sử dụng để tạo một instance của một object. Để tạo một object, toán tử mới được theo sau bởi phương thức constructor.

Sau đây là cú pháp để xác định một object.

var obj\_name = new Object();

obj\_name.property = value;

// Hoặc

obj\_name["key"] = value

Sau đây là cú pháp để truy cập thuộc tính.

Object\_name.property\_key

// Hoặc

Object\_name["property\_key"]

Ví dụ

var myTV = new Object();

myTV.make = "Sony";

myTV.model = "Bravia";

myTV.year = 2016;

console.log(myTV ["make"])

console.log(myTV ["model"])

console.log(myTV ["year"])

**Phần 3: Khai báo class trong ES6 (Tác giả Lê Đình Cường)**

Lúc trước thì javascript thì chúng ta tạo class bằng fuction nhưng đến es6 thì chúng ta đã có class đúng nghĩa.

**Cách tạo một class**

class **TenClass** {

}

VD: Chúng ta tạo một class Person

class **Person** {

}

**Tạo constructor (Phương thức khởi tạo cho class)**

Trong class ta sẽ viết như sau

class **Employee** {

*// Day là buoc chúng ta khởi tạo một constructor với tham số truyền vào là name và age*

constructor (name, age) {

this.name = name;

this.age = age;

}

};

**Cách khai báo getter và setter**

class **Employee** {

*//Dưới đây là cách chúng ta tạo setter cho biến name*

set **employeeName** (name) {

this.name = name;

}

*// Đây là cách chúng ta tạo phương thức getter cho biến name*

get **employeeName** () {

return this.name;

}

};

*//khởi tạo đối tượng Employee*

var employee = new **Employee**();

*//setter*

employee.employeeName = "Le Dinh Cuong"

*//getter*

employee.employeeName;

*//kết quả: Le Dinh Cuong*

**Kế thừa trong class**

*// Lop A kế thừa các thuộc tính của lớp B*

class **A** extends **B** {

*//code*

}

Cách gọi phương thức của lớp cha trong lớp con thì tại một phương thức nào đó của lớp con ta gọi theo cấu trúc như sau: super.tenPhuongThuc();

*// Class cha*

class **Employee** {

**getClassName** () {

return "Class Employee";

}

};

*// Class con MaleEmployee kế thừa class cha là Employee*

class **MaleEmployee** extends **Employee** {

**getClassName** () {

*// Ta gọi phương thức getClassName trong cha của nó như sau*

return "Class MaleEmployee - " + super.**getClassName**();

}

**classClassName** () {

return super.**getClassName**();

}

}

**Cách tạo một phương thức tĩnh trong class**

Để tạo một phương thức tĩnh trong class thì ta khai báo trước phương thức có từ khóa static.

Cấu trúc

static tenPhuongThuc () {…}

VD

class **Employee** {

static **defaultEmployee** () {

return "Đây là phương thức defaultEmployee";

}

};

*// Gọi phương thức tĩnh*

className.**staticMethodName**();

**Cách viết một phương thức trong class**

tenPhuongThuc() {…};

**Cách để khởi tạo một class**

new TenClass;

hoặc

new TenClass();