**CHƯƠNG 2:** **Kiến thức cơ bản về ES6**

1. Biến Let, Var, Const trong ES6
2. Arrow Function
3. Kiểu dữ liệu Object
4. Khai báo Class trong ES6

2.1Biến Let, Var, Conts trong ES6

* 1. Arrow Function

## Arrow function - còn được gọi là "fat arrow", là cú pháp được mượn từ CoffeeScript (một ngôn ngữ chuyển tiếp), cú pháp này là cách ngắn gọn hơn dùng để viết function. Ở đây sử dụng kí tự =>, trông giống như một mũi tên "béo". Arrow function là một hàm vô danh và nó thay đổi cách this bind đến function. Arrow function làm code của ta trông ngắn gọn hơn, giúp đơn giản hóa function scoping cũng như từ khóa this

## Cách dùng arrow function

## Trong trường hợp nhiều tham số

// (param1, param2, paramN) => expression

// ES5

var multiply = function(x, y) {

return x \* y;

};

// ES6

var multiply = (x, y) => { return x \* y };

Ví dụ trên cho cùng một kết quả, tuy nhiên cú pháp với arrow function tốn ít dòng mã hơn. Trong trường hợp chỉ có một biểu thức thì không cần tới dấu ngoặc nhọn: Ví dụ trên có thể viết lại như sau:

var multiply = (x, y) => x \* y ;

### Trong trường hợp 1 tham số

Dấu ngoặc đơn là optional khi chỉ có một tham số

//ES5

var phraseSplitterEs5 = function phraseSplitter(phrase) {

return phrase.split(' ');

};

//ES6

var phraseSplitterEs6 = phrase => phrase.split(" ");

console.log(phraseSplitterEs6("ES6 Awesomeness")); // ["ES6", "Awesomeness"]

### Trong trường hợp không có tham số

Dấu ngoặc đơn là bắt buộc khi không có tham số.

//ES6

var docLogEs6 = () => { console.log(document); }

docLogEs6(); // #document... <html> ….

## Khi nào thì nên sử dụng arrow function

Một usecase phổ biến áp dung arrow function là thao tác mảng, thông thường là khi dùng maphoặc reduce. Có 1 mảng như sau:

var smartPhones = [

{ name: 'iphone', price: 649 },

{ name: 'Galaxy S6', price: 576 },

{ name: 'Galaxy Note 5', price: 489 }

];

Để lấy 1 mảng toàn tên các smart phones, ta thực hiện:

// ES5

console.log(smartPhones.map(function(smartPhone){

return smartPhone.price;

}); //

Với arrow function, có thể viết rút gọn lại thành

// ES6

console.log(smartPhones.map(smartPhone => smartPhone.price)); // [649, 576, 489]

1. Khai báo Class trong ES6

Trong ES6 đã hỗ trợ chúng ta khai báo một đối tượng theo chuẩn OOP, bằng cách sử dụng từ khóa class.

**VD**: Đối với VD trên chúng ta có thể chuyển sang dạng ES6 thành như sau:

var employee = {

name: null,

age: null,

setName: function (name) {

this.name = name;

},

getName: function () {

return this.name;

},

setAge: function (age) {

this.age = age;

},

getAge: function () {

return this.age;

}

};

Với ES6 thì bạn không thể khai báo các thuộc tính như bình thường được mà chỉ có thể gán nó vào các phương thức trong đối tượng được thôi.

Để khởi tạo đối tượng được khai báo theo chuẩn ES6 thì các bạn sử dụng từ khóa new với cú pháp như sau:

new ClassName;

//hoặc

new ClassName();

Trong đó className là tên của đối tượng mà các bạn muốn khởi tạo.

**VD**: Mình sẽ khởi tạo đối tượng Employee được khai báo ở ví dụ trên.

new Employee();

Và với ES6, nó cũng hỗ trợ chúng ta một phương thức đặc biệt mà bất kỳ ngôn ngữ lập trình nào cũng có đối với class đó là constructor - phương thức khởi tạo. constructor trong ES6 cũng có tác dụng tương tự, nó sẽ tự động được gọi khi đối tượng được khởi tạo.

Để khai báo constructor trong ES6 thì các bạn chỉ cần khai báo một phương thức có tên là constructor.

**VD**: khai báo constuctor cho đối tượng Employee ở trên.

class Employee {

constructor (name, age) {

this.name = name;

this.age = age;

}

setName (name) {

this.name = name;

}

getName () {

return this.name;

}

setAge (age) {

this.age = age;

}

getAge () {

return this.age;

}

};

Như mình cũng đã nói ở trên thì với kiểu khai báo class trong ES6 chúng ta không thể khai báo trực tiếp thuộc tính cho nó được mà phải khởi tạo qua các phương thức và thường thì chúng ta sẽ đặt nó ở trong constructor luôn.

Lúc này khi khởi tạo đối tượng chúng ta có thể truyền luôn tham số cho nó như các ngôn ngữ khác.

**VD**:

new Employee("Vũ Thanh Tài", 22);

Bạn nào muốn kiểm chứng có thể console.log() đối tượng này ra để xem class sẽ xuất hiện ra hai thuộc tính nam và age với giá trị như khởi tạo.

## 2, Class Inheritance.

Đã theo chuẩn OOP-style thì phải đầy đủ đúng không mọi người? Với ES6, nó cũng đã cung cấp cho chúng ta sử dụng từ khóa extends để kế thừa từ đối tượng khác.

**Cú Pháp**:

class A extends B {

//code

}

**Trong đó**: A là class đang được khai báo, và nó kế thừa lại các thuộc tính và phương thức từ class B.

**VD**: Mình sẽ viết một đối tượng MaleEmployee kế thừa từ class Employee ở trên.

class MaleEmployee extends Employee {

constructor (name,age,wifeName) {

super(name,age);

this.wifeName = wifeName;

}

setWifeName (wifeName) {

this.setWifeName;

}

getWifeName () {

return this.wifeName;

}

}

Trong ví dụ này mình có sử dụng super() - viết với cú pháp như này thì là gọi lại constructor của lớp cha, còn chi tiết về nó phần dưới mình sẽ nói rõ hơn.

## 3, Base class access.

Trong ES6, để thực hiện gọi các phương thức trong lớp cha khi đang ở lớp con, mà phương thức đó đã bị rewritetrong lớp con rồi thì các bạn sử dụng keyword super với cú pháp như sau:

super.methodName();

Trong đó, methodName là phương thức của lớp cha mà bạn muốn gọi.

**VD**:

class Employee {

getClassName () {

return "Class Employee";

}

};

class MaleEmployee extends Employee {

getClassName () {

return "Class MaleEmployee";

}

classClassName () {

return super.getClassName();

}

}

var employee = new MaleEmployee();

console.log(employee.classClassName());

//kết quả: Class Employee

Và bạn cũng có thể gọi lại phương thức của lớp cha khi đang đứng trong phương thức đó ở lớp con (trong phương thức rewrite chính nó).

**VD**:

class Employee {

getClassName () {

return "Class Employee";

}

};

class MaleEmployee extends Employee {

getClassName () {

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

}

classClassName () {

return super.getClassName();

}

}

var employee = new MaleEmployee();

console.log(employee.getClassName());

//kết quả: main.js:15 Class MaleEmployee - Class Employee

## 4, Static Members.

Trong ES6 cũng hỗ trợ chúng ta khai báo các thành phần tĩnh cho đối tượng, bằng cách sử dụng từ khóa static ở trước tên phương thức.

**VD**:

class Employee {

static defaultEmployee () {

return "Đây là phương thức defaultEmployee";

}

};

Và khi một phương thức được khai báo là static methods thì chúng ta sẽ không thể gọi phương thức đó một cách thông thường được nữa, mà chúng ta sẽ gọi theo cú pháp sau:

className.staticMethodName();

**Trong đó**:

* className là tên của class chứa static methods mà bạn muốn gọi..
* staticMethodName là tên của static methods mà bạn muốn gọi.

**VD**: Mình sẽ gọi phương thức defaultEmployee của class Employee ở trên.

console.log(Employee.defaultEmployee());

//kết quả: Đây là phương thức defaultEmployee

## 5, Setter and Getter.

Nếu như bạn đã học qua [hướng đối tượng](https://toidicode.com/php-huong-doi-tuong-oop-32" \t "_blank) rồi thì chắc hẳn bạn sẽ không còn gì lạ lẫm với các setter và getter nữa. Và trong ES6 cũng hỗ trợ chúng ta thiết lập các setter và getter cho các thuộc tính.

*setter và getter là những phương thức đặc biệt (magic method) chúng được gọi khi chúng ta tác động lên các thuộc tính trong đối tượng (Mình giải thích qua loa cho những ai chưa biết thôi còn chi tiết các bạn chịu khó google hoặc comment mình sẽ khái niệm chi tiết lại).*

Để khai báo các setter và getter trong ES6 chúng ta sử dụng keyword set và get trước tên các phương thức mà bạn muốn thiết lập nó là setter hoặc getter.

**VD**: Mình sẽ chuyển đổi các phương thức setName, getName, setAge, getAge trong đối tượng employee ở trên về dạng setter getter.

class Employee {

constructor (name, age) {

this.name = name;

this.age = age;

}

set employeeName (name) {

this.name = name;

}

get employeeName () {

return this.name;

}

set employeeAge (age) {

this.age = age;

}

get employeeAge () {

return this.age;

}

};

Lúc này chúng ta sẽ sử dụng setter getter như làm với một thuộc tính thông thường.

**VD**:

//khởi tạo đối tượng

var employee = new Employee();

//setter

employee.employeeName = "Vũ Thanh Tài"

//getter

employee.employeeName;

//kết quả: Vũ Thanh Tài

Hoặc bạn cũng có thể thực hiện setter, getter bằng cách truy cập trực tiếp vào class mà không cần khởi tạo.

//setter

Employee.employeeName = "Vũ Thanh Tài"

//getter

Employee.employeeName;

//kết quả: Vũ Thanh Tài