Nhóm 4.

Cách xử lý mảng trong ES6

1. Khai báo và khởi tạo mảng.

Để khai báo mảng sử dụng câu lệnh sau :

var array\_name; //declaration

array\_name = [val1,val2,valn..] //initialization

OR

var array\_name = [val1,val2…valn]

Cặp [] được gọi là thứ nguyên của mảng.

1. Truy cập các phần tử mảng.

Truy cập bằng chỉ mục của mảng

Cú pháp

array\_name[subscript]

ví dụ:

mảng 1 chiều:

var alphas;

alphas = ["1","2","3","4"]

console.log(alphas[0]);

console.log(alphas[1]);

kết quả :

1

2

var nums = [1,2,3,3]

console.log(nums[0]);

console.log(nums[1]);

console.log(nums[2]);

console.log(nums[3]);

kết quả :

1

2

3

3

1. Đối tượng mảng

Một mảng cũng có thể được tạo ra bằng cách sử dụng đối tượng mảng. Các nhà xây dựng mảng có thể được thông qua như sau :

* Một giá trị số đại diện cho kích thước của mảng hoặc.
* Danh sách các giá trị được phân cách bằng dấu phẩy.
* var arr\_names = new Array(4)
* for(var i = 0;i<arr\_names.length;i++) {
* arr\_names[i] = i \* 2
* console.log(arr\_names[i])
* }

var names = new Array("Mary","Tom","Jack","Jill")

for(var i = 0;i<names.length;i++) {

console.log(names[i])

}

## Các phương pháp mảng

[**concat ()**](https://www.tutorialspoint.com/es6/es6_array_method_concat.htm)

## dùng để kết nối 2 hay nhiều mảng với nhau. Phương thức này không làm thay đổi các mảng đã có mà thay vào đó sẽ trả về 1 mảng mới.

## [every()](https://www.tutorialspoint.com/es6/es6_array_method_every.htm)

## trả về giá trị nếu đùng điều kiện.

## [filter()](https://www.tutorialspoint.com/es6/es6_array_method_filter.htm)

## dùng để tạo một mảng mới với tất cả các phần tử thỏa điều kiện của một hàm test.

## [forEach()](https://www.tutorialspoint.com/es6/es6_array_method_foreach.htm)

## duyệt qua từng phần tử của mảng.

## Phương pháp mảng

### **Array.prototype.find**

## cho phép bạn lặp qua một mảng và lấy lại phần tử đầu tiên làm cho hàm gọi lại đã cho trả về true. Khi một phần tử đã được tìm thấy, hàm này sẽ trả về ngay lập tức.

var numbers = [1, 2, 3];

var oddNumber = numbers.find((x) => x % 2 == 1);

console.log(oddNumber); // 1

### **Array.prototype.findIndex**

## hoạt động tương tự như tìm, nhưng thay vì trả về phần tử đã khớp, nó trả về chỉ mục của phần tử đó.

var numbers = [1, 2, 3];

var oddNumber = numbers.findIndex((x) => x % 2 == 1);

console.log(oddNumber); // 0

### **Array.prototype.entries**

## là một hàm trả về một Iterator mảng có thể được sử dụng để lặp qua các khóa và giá trị của mảng. Các mục sẽ trả về một mảng các mảng, trong đó mỗi mảng con là một mảng của [chỉ mục, giá trị].

var numbers = [1, 2, 3];

var val = numbers.entries();

console.log(val.next().value);

console.log(val.next().value);

console.log(val.next().value);

### **Array.from**

cho phép tạo một mảng mới từ một mảng như đối tượng. Chức năng cơ bản của Array.from () là chuyển đổi hai loại giá trị thành mảng -

* Giá trị giống như mảng.
* Các giá trị lặp lại như Set và Map.
* "use strict"
* for (let i of Array.from('hello')) {
* console.log(i)
* }
* Đầu ra sau được hiển thị khi thực hiện thành công mã trên.
* h
* e
* l
* l
* o

### **Array.prototype.keys ()**

Hàm này trả về các chỉ mục mảng.

Thí dụ

console.log(Array.from(['a', 'b'].keys()))

Đầu ra sau được hiển thị khi thực hiện thành công mã trên.

[ 0, 1 ]

Người ta có thể sử dụng cho ... trong vòng lặp để đi qua một mảng.

"use strict"

var nums = [1001,1002,1003,1004]

for(let j in nums) {

console.log(nums[j])

}

Vòng lặp thực hiện truyền tải mảng dựa trên chỉ mục. Đầu ra sau được hiển thị khi thực hiện thành công mã trên.

1001

1002

1003

1004