[Bài đọc] Mảng cơ bản

1. Mảng là gì?

Mảng là một loại biến đặc biệt, được dùng để lưu trữ nhiều giá trị.

Trong các trường hợp mà chúng ta cần lưu trữ một danh sách các giá trị, chẳng hạn như danh sách khách hàng, danh sách sản phẩm, danh sách cấu hình... thì mảng là một lựa chọn tốt.

Chẳng hạn, nếu chúng ta sử dụng biến bình thường để lưu 3 tên xe thì đoạn mã sẽ như sau:

```
let car1 = "Toyota";
let car2 = "Subaru";
let car3 = "BMW";
```

Thử tưởng tượng, nếu chúng ta có danh sách 100 chiếc xe thì sẽ làm sao? Chẳng lẽ lại đi khai báo 100 biến tương ứng.

Trong khi, nếu chúng ta sử dụng mảng thì nó sẽ ngắn gọn là:

```
let cars = ["Toyota", "Subaru", "BMW"];
```

2. Các thành phần của một mảng

Khi khai báo và làm việc với mảng, chúng ta cần biết các thành phần sau:

- **Tên mảng (name)**: Phải tuân thủ theo quy tắc đặt tên của biến
- **Độ dài của mảng (length)**: Là số lượng giá trị mà mảng có thể chứa.

```
>cars.length
// 3
```

- **Phần tử (item)**: Là một giá trị trong mảng

- **Chỉ số của phần tử (item index)**: Là vị trí của phần tử trong mảng. Chỉ số đầu tiên là 0, tiếp sau là 1, 2, 3... cho đến hết. Chỉ số của phần tử cuối cùng của mảng sẽ là length - 1. Trong đó length là độ dài của mảng. Chẳng hạn, nếu mảng có 6 phần tử thì chỉ số của phần tử cuối cùng sẽ là 5.

```
>cars[0]
// Toyota
>cars[2]
//BMW
```

3. Cú pháp khai báo mảng

Có một số cách khác nhau để khai báo mảng.

Cách 1: Sử dụng dấu ngoặc vuông ([]) để khai báo mảng:

```
let arr = [element1, element2, element3];
```

Ví dụ:

```
let arr = ["Toyota", "Subaru", "BMW"];
```

Lưu ý: Dấu cách, hoặc dấu xuống dòng không có ảnh hưởng gì tới việc khai báo mảng. Chẳng hạn, chúng ta có thể khai báo trên nhiều dòng như sau:

```
let arr = [
   "Toyota",
   "Subaru",
   "BMW"
];
```

Cách 2: Sử dụng từ khoá new:

```
let arr = new Array("Toyota", "Subaru", "BMW");
```

Việc sử dụng từ khoá new hay dấu ngoặc vuông đều cho kết quả như nhau. Thông thường thì cú pháp sử dụng dấu ngoặc vông được sử dụng nhiều hơn do ngắn gọn, dễ đọc.

4. Truy xuất một phần tử của mảng

Để truy xuất đến một phần tử của mảng thì chúng ta sử dụng tên mảng và chỉ số của phần tử đó.

Ví dụ, chúng ta truy xuất đến phần tử đầu tiên của mảng cars như sau:

```
let toy = cars[0];
```

Chúng ta cũng có thể sử dụng cách tương tự để gán giá trị cho các phần tử của mảng.

Ví dụ, chúng ta gán giá trị mới cho phần tử đầu tiên của mảng cars như sau:

```
cars[3] = "Huyndai";
```

5. Độ dài của mảng

Chúng ta sử dụng thuộc tính length để biết độ dài của mảng.

Ví dụ:

```
let countOfCars = cars.length; // 4
```

Trong ví dụ trên, độ dài của mảng sẽ là 4.

6. Thêm phần tử vào mảng

Hàm push() sẽ thêm một phần tử mới vào phần cuối của mảng.

Ví dụ:

```
cars.push("Kia");
console.log(cars[cars.length - 1]); // "Kia
```

Hàm unshift() sẽ thêm một phần tử mới và phần đầu của mảng.

Ví dụ:

```
cars.unshift("Ferari");
console.log(cars[0]) // "Ferari
```

7. Xoá phần tử của mảng

Hàm pop() sẽ lấy đi phần tử cuối cùng của mảng.

Ví dụ:

```
cars.length; // 6
let last = cars.pop(); // Kia
cars.length; // 5
```

Hàm shift() sẽ lấy đi phần tử đầu tiên của mảng.

Ví dụ:

```
cars.length; // 5
let last = cars.shift(); // "Ferari"
cars.length; // 4
```

8. Sắp xếp mảng

Hàm sort() sẽ giúp sắp xếp các phần tử của mảng theo một trật tự nhất định.

Ví dụ:

```
let arr = ["Toyota", "Subaru", "BMW"];
arr.sort(); // [ "Toyota", "Subaru", "BMW" ]
```

Chúng ta cũng có thể đảo ngược trật tự của một mảng bằng cách sử dụng hàm reverse().

Ví dụ:

```
let arr = ["Toyota", "Subaru", "BMW"];
arr.sort(); // [ "BMW", "Subaru", "Toyota" ]
arr.reverse(); // [ "Toyota", "Subaru", "BMW" ]
```

Theo mặc định, phương thức sort() sẽ so sánh các phần tử theo trật tự của các ký tự trong bảng chữ cái.

Chẳng hạn, phần tử "Apple" sẽ được đưa lên trước phần tử "Banana".

Tuy nhiên, điều này sẽ gây sai sót khi so sánh các chữ số, chẳng hạn, số 100 sẽ được đưa lên trước số 25 (bởi vì số 1 đứng trước số 2 trong bảng chữ cái). Trong trường hợp này, chúng ta cung cấp cho hàm sort() một hàm so sánh, để nó thực hiện đúng chức năng của mình.

Ví dụ:

```
let points = [40, 100, 1, 5, 25, 10];
points.sort(function(a, b) {
   return a - b;
}) // [ 1, 5, 10, 25, 40, 100 ]
points.sort(function(a, b) {
   return b - a;
}) // [ 100, 40, 25, 10, 5, 1 ]
```

9. Lưu ý về hàm so sánh

Hàm so sánh là một hàm có mục đích giúp so sánh 2 phần tử.

Hàm so sánh có 2 tham số

Hàm so sánh trả về giá trị kiểu số: âm (nhỏ hơn 0), dương (lớn hơn 0) và 0.

- âm có nghĩa là phần tử đầu nhỏ hơn phần tử sau
- dương có nghĩa là phần tử đầu lớn hơn phần tử sau
- 0 có nghĩa là 2 phần tử bằng nhau