Buổi 7

Mảng trong JavaScript

I. Khai báo và gán giá trị cho mảng

Mảng (*array*) trong JS có thể được khai báo theo 3 cách:

- Cách 1 – Regular Array:

```
var monHoc = new Array();
monHoc[0] = "Toán";
monHoc[1] = "Lý";
monHoc[2] = "Hóa";
```

- Cách 2 – Condensed Array:

```
var monHoc = new Array("Toán", "Lý", "Hóa");
```

- Cách 3 – Literal Array:

```
var monHoc = ["Toán", "Lý", "Hóa"];
```

Tuy nhiên, để tối ưu về mặt tốc độ thực thi cũng như để code dễ đọc hiểu hơn, chúng ta sử dụng cách 3.

II. Duyệt mảng

Để truy cập các phần tử của mảng, chúng ta sử dụng chỉ số (*index*) bắt đầu từ 0.

JS hỗ trợ 3 cách duyệt mảng:

- Cách 1 – Vòng lặp `for`:

```
for (var i = 0; i < monHoc.length; i++) {
    // Thao tác trên monHoc[i]
}
```

- Cách 2 – Vòng lặp `for/in`:

```
for (i in monHoc) {
    // Thao tác trên monHoc[i]
}
```

- Cách 3 – Vòng lặp `for/of`:

```
for (i of monHoc) {
    // Thao tác trên phần tử i
}
```

III. Thêm và xóa phần tử trong mảng

JS hỗ trợ phương thức `push()` và `pop()` để thêm và xóa phần tử ở phía cuối mảng.

Ví dụ:

```
monHoc.push("Tin học");
var x = monHoc.pop();       // Tại đây x có giá trị là "Tin học"
```

JS hỗ trợ phương thức `unshift()` và `shift()` để thêm và xóa phần tử ở phía đầu mảng.

Ví dụ:

```
monHoc.unshift("Anh Văn");
var x = monHoc.shift();     // Tại đây x có giá trị là "Anh Văn"
```

Phương thức `push()` và `unshift()` sẽ trả về số lượng phần tử của mảng sau khi thêm.

Phương thức `pop()` và `shift()` sẽ trả về giá trị của phần tử vừa bị xóa.

Ngoài ra, có thể thêm phần tử bằng chỉ số trực tiếp. Tuy nhiên, cách này có thể tạo ra các phần tử rỗng ở giữa mảng:

```
monHoc[10] = "Văn học";
```

IV. Các thuộc tính và phương thức của mảng

Thuộc tính `length` cho biết số lượng phần tử của mảng. Phần tử cuối cùng của mảng `monHoc` luôn là `monHoc[monHoc.length - 1]`.

Phương thức `toString()` và `join()` dùng để chuyển mảng về kiểu chuỗi:

`monHoc.toString()` → Toán,Lý,Hóa

`monHoc.join(" - ")` → Toán – Lý – Hóa

Phương thức `reverse()` dùng để đảo ngược mảng.

Phương thức `sort()` dùng để sắp xếp mảng tăng dần theo ABC. Kết hợp `sort()` và `reverse()` để sắp xếp mảng giảm dần.

Tuy nhiên, khi dùng `sort()` để sắp xếp số có thể sẽ cho kết quả sai vì phương thức này sẽ xem số như chuỗi, do đó, số 25 sẽ lớn hơn 100 (vì chuỗi "25" lớn hơn "100"). Để sắp xếp số của mảng `arr`, chúng ta viết phương thức `sort()` như sau:

- Sắp xếp số tăng dần: `arr.sort(function(a, b) { return a - b } );`
- Sắp xếp số giảm dần: `arr.sort(function(a, b) { return b - a } );`

V. Các phương thức tuần tự trong mảng

Mảng trong JS cung cấp một số phương thức giúp thực hiện một thao tác cho mỗi phần tử của mảng, thay vì phải dùng vòng lặp để duyệt từng phần tử. Trong khuôn khổ tài liệu buổi 7, chúng ta tạm gọi đó là các phương thức tuần tự[1].

Giả sử có mảng sau:

```
var arr = [9, 6, 3, 2, 5, 7, 1, 2];
```

1. Tìm kiếm phần tử

Phương thức `indexOf()` và `lastIndexOf()` cho biết vị trí đầu tiên và cuối cùng của 1 phần tử nào đó trong mảng:

`arr.indexOf(2)` → Trả về vị trí 3

`arr.lastIndexOf(2)` → Trả về vị trí 7

Phương thức `findIndex()` cho biết vị trí của phần tử đầu tiên thỏa mãn 1 điều kiệu nào đó:

```
function NhoHon5(value) {
    return value < 5;
}
arr.findIndex(NhoHon5); → Trả về vị trí 2
```

Phương thức `find()` cho biết giá trị của phần tử đầu tiên thỏa mãn 1 điều kiện nào đó:

`arr.find(NhoHon5);` → Trả về giá trị 3

2. Kiểm tra phần tử

Phương thức `every()` kiểm tra xem có phải tất cả phần tử đều thỏa mãn 1 điều kiện nào đó hay không:

`arr.every(NhoHon5);` → Trả về `false`

[1] Xem thêm tại: https://www.w3schools.com/js/js_array_iteration.asp

Phương thức `some()` kiểm tra xem có tồn tại phần tử thỏa mãn 1 điều kiện nào đó hay không:

`arr.some(NhoHon5);` → Trả về `true`

Phương thức `filter()` lọc ra các phần tử thỏa mãn 1 điều kiện nào đó:

`arr.filter(NhoHon5);` → Trả về mảng mới `[3, 2, 1, 2]`

3. Thay đổi phần tử

Phương thức `forEach()` thực hiện 1 hàm nào đó khi xét từng phần tử:

```
var tong = 0;
function TinhTong(value) {
	tong = tong + value;
}
arr.forEach(TinhTong);
```

→ Trả về 35

Phương thức `map()` thay đổi từng phần tử theo 1 hàm nào đó:

```
function NhanDoi(value) {
	return value * 2;
}
arr.map(NhanDoi);
```

→ Trả về mảng mới `[18, 12, 6, 4, 10, 14, 2, 4]`

VI. Bài tập

Thực hiện các bài tập dưới đây, mỗi bài tập nằm trong 1 trang web riêng.

Yêu cầu chung: Thiết kế 1 trang web gồm 1 textbox cho phép người dùng nhập 1 số nguyên. Cứ mỗi khi người dùng nhấn nút *Thêm*, thêm số đó vào 1 mảng, hiển thị mảng ra màn hình.

1. Cho biết mảng có số âm hay không.
2. Cho biết mảng có phải chỉ gồm số nguyên tố hay không (Gợi ý: dùng phương thức `every()`).
3. Cho biết danh sách các số nguyên tố trong mảng (Gợi ý: dùng phương thức `filter()`).