MẢNG TRONG JAVASCRIPT

Hướng dẫn chi tiết và ví dụ thực tế

Ngày: 1 tháng 6, 2025

---

1. Giới thiệu về Mảng

Mảng (Array) là một cấu trúc dữ liệu quan trọng trong JavaScript, cho phép lưu trữ nhiều giá trị trong một biến duy nhất. Mảng trong JavaScript là động, có nghĩa là bạn có thể thay đổi kích thước và thêm/xóa phần tử bất kỳ lúc nào.

Lưu ý quan trọng:

Mảng trong JavaScript có thể chứa các kiểu dữ liệu khác nhau trong cùng một mảng.

2. Cách tạo mảng

2.1. Sử dụng Array Literal (Khuyến nghị)

```
// Tạo mảng rỗng let emptyArray = []; // Tạo mảng với các phần tử let fruits = ['apple', 'banana', 'orange']; let numbers = [1, 2, 3, 4, 5]; let mixed = ['hello', 42, true, null];
```

2.2. Sử dụng Constructor Array

```
// Tạo mảng rỗng let emptyArray = new Array(); // Tạo mảng với kích thước cố định let fixedArray = new Array(5); // Mảng 5 phần tử undefined // Tạo mảng với các phần tử let fruits = new Array('apple', 'banana', 'orange');
```

Ví dụ thực tế:

```
// Danh sách sinh viên trong lớp let students = ['Nguyễn Văn A', 'Trần Thị B', 'Lê Văn C']; // Danh sách điểm số let scores = [8.5, 7.2, 9.0,
```

```
6.8, 8.9]; // Thông tin hỗn hợp let studentInfo = ['Nguyễn Văn A', 20, 'Nam', 8.5];
```

3. Truy cập phần tử mảng

Mảng trong JavaScript sử dụng chỉ số (index) để truy cập phần tử, bắt đầu từ 0.

```
let fruits = ['apple', 'banana', 'orange', 'grape']; console.log(fruits[0]); // 'apple' console.log(fruits[1]); // 'banana' console.log(fruits[2]); // 'orange' console.log(fruits[3]); // 'grape' // Truy cập phần tử cuối cùng console.log(fruits[fruits.length - 1]); // 'grape'
```

Ví dụ với mảng điểm số:

```
let scores = [8.5, 7.2, 9.0, 6.8, 8.9]; console.log('Điểm bài kiểm tra 1:', scores[0]); // 8.5 console.log('Điểm bài kiểm tra 2:', scores[1]); // 7.2 console.log('Điểm cao nhất:', Math.max(...scores)); // 9.0
```

4. Thuộc tính và phương thức cơ bản

4.1. Thuộc tính length

```
let fruits = ['apple', 'banana', 'orange']; console.log(fruits.length); // 3 // Thay đổi độ dài mảng fruits.length = 2; console.log(fruits); // ['apple', 'banana']
```

4.2. Thêm phần tử

```
let fruits = ['apple', 'banana']; // Thêm vào cuối mảng fruits.push('orange'); console.log(fruits); // ['apple', 'banana', 'orange'] // Thêm vào đầu mảng fruits.unshift('grape'); console.log(fruits); // ['grape', 'apple', 'banana', 'orange'] // Thêm vào vị trí cụ thể
```

```
fruits.splice(2, 0, 'kiwi'); console.log(fruits); // ['grape', 'apple', 'kiwi', 'banana', 'orange']
```

4.3. Xóa phần tử

```
let fruits = ['apple', 'banana', 'orange', 'grape']; // Xóa phần tử cuối let lastFruit = fruits.pop(); console.log(lastFruit); // 'grape' console.log(fruits); // ['apple', 'banana', 'orange'] // Xóa phần tử đầu let firstFruit = fruits.shift(); console.log(firstFruit); // 'apple' console.log(fruits); // ['banana', 'orange'] // Xóa phần tử tại vị trí cụ thể fruits.splice(1, 1); // Xóa 1 phần tử tại index 1 console.log(fruits); // ['banana']
```

5. Duyệt mảng

5.1. Vòng lặp for truyền thống

```
let numbers = [1, 2, 3, 4, 5]; for (let i = 0; i < numbers.length; i++) { console.log(`Phần tử ${i}: ${numbers[i]}`); }
```

5.2. Vòng lặp for...of

```
let fruits = ['apple', 'banana', 'orange']; for (let fruit of fruits) { console.log(fruit); }
```

5.3. Phương thức forEach

```
let numbers = [1, 2, 3, 4, 5]; numbers.forEach(function(number, index) { console.log(`Index ${index}: ${number}`); }); // Arrow function numbers.forEach((number, index) => { console.log(`Index ${index}: ${number}`); });
```

Ví dụ tính tổng điểm:

```
let scores = [8.5, 7.2, 9.0, 6.8, 8.9]; let total = 0; scores.forEach(score => { total += score; }); let average = total /
```

```
scores.length; console.log(`Điểm trung bình: ${average.toFixed(2)}`); // 8.08
```

6. Các phương thức quan trọng

6.1. Phương thức map()

Tạo mảng mới bằng cách biến đổi từng phần tử.

```
let numbers = [1, 2, 3, 4, 5]; let doubled = numbers.map(num => num * 2); console.log(doubled); // [2, 4, 6, 8, 10] let prices = [100, 200, 300]; let pricesWithTax = prices.map(price => price * 1.1); console.log(pricesWithTax); // [110, 220, 330]
```

6.2. Phương thức filter()

Tạo mảng mới chứa các phần tử thỏa mãn điều kiện.

```
let numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]; let evenNumbers = numbers.filter(num => num % 2 === 0); console.log(evenNumbers); // [2, 4, 6, 8, 10] let students = [ {name: 'A', score: 8.5}, {name: 'B', score: 6.2}, {name: 'C', score: 9.1}, {name: 'D', score: 7.8} ]; let passedStudents = students.filter(student => student.score >= 8.0); console.log(passedStudents);
```

6.3. Phương thức find() và findIndex()

```
let products = [ {id: 1, name: 'Laptop', price: 1000}, {id: 2, name: 'Phone', price: 500}, {id: 3, name: 'Tablet', price: 300} ]; // Tìm sản phẩm let laptop = products.find(product => product.name === 'Laptop'); console.log(laptop); // {id: 1, name: 'Laptop', price: 1000} // Tìm index let phoneIndex = products.findIndex(product => product.name === 'Phone'); console.log(phoneIndex); // 1
```

6.4. Phương thức reduce()

Giảm mảng thành một giá trị duy nhất.

```
let numbers = [1, 2, 3, 4, 5]; let sum = numbers.reduce((total, current) => total + current, 0); console.log(sum); // 15 let products = [ {name: 'Laptop', price: 1000}, {name: 'Phone', price: 500}, {name: 'Tablet', price: 300} ]; let totalPrice = products.reduce((total, product) => total + product.price, 0); console.log(totalPrice); // 1800
```

7. Ví dụ thực tế

Quản lý danh sách sinh viên:

```
let students = [ {id: 1, name: 'Nguyễn Văn A', age: 20, scores: [8, 7, 9]}, {id: 2, name: 'Trần Thị B', age: 19, scores: [7, 8, 8]}, {id: 3, name: 'Lê Văn C', age: 21, scores: [9, 9, 8]} ]; // Tính điểm trung bình cho mỗi sinh viên students.forEach(student => { let average = student.scores.reduce((sum, score) => sum + score, 0) / student.scores.length; student.average = Math.round(average * 100) / 100; }); console.log(students); // Tìm sinh viên có điểm cao nhất let topStudent = students.reduce((prev, current) => prev.average > current.average ? prev : current ); console.log('Sinh viên xuất sắc nhất:', topStudent.name);
```

Xử lý giỏ hàng:

```
let cart = [ {name: 'Áo sơ mi', price: 200000, quantity: 2}, {name: 'Quần jean', price: 350000, quantity: 1}, {name: 'Giày thể thao', price: 800000, quantity: 1} ]; // Tính tổng tiền let totalAmount = cart.reduce((total, item) => { return total + (item.price * item.quantity); }, 0); // Format tiền tệ let formattedTotal = new Intl.NumberFormat('vi-VN', { style: 'currency', currency: 'VND' }).format(totalAmount); console.log('Tổng tiền:', formattedTotal); // 1.350.000 ₫ // Hiển thị chi tiết cart.forEach(item => { console.log(`${item.name}: ${item.quantity} x ${item.price.toLocaleString('vi-VN')}đ`); });
```

8. Bảng tóm tắt các phương thức

Phương thức	Mô tả	Ví dụ
push()	Thêm phần tử vào cuối mảng	arr.push(5)
pop()	Xóa phần tử cuối mảng	arr.pop()
shift()	Xóa phần tử đầu mảng	arr.shift()
unshift()	Thêm phần tử vào đầu mảng	arr.unshift(1)
splice()	Thêm/xóa phần tử tại vị trí	arr.splice(1, 2, 'new')
map()	Tạo mảng mới sau biến đổi	arr.map(x => x * 2)
filter()	Lọc phần tử theo điều kiện	arr.filter(x => x > 5)
find()	Tìm phần tử đầu tiên	arr.find(x => x > 5)
reduce()	Giảm mảng thành một giá trị	arr.reduce((a, b) => a + b)

9. Lưu ý quan trọng

- Mảng trong JavaScript là reference type, khi gán một mảng cho biến khác, cả hai sẽ trỏ đến cùng một vùng nhớ.
- Sử dụng spread operator (...) để copy mảng: `let newArr = [...oldArr]`
- Các phương thức như map(), filter(), reduce() không thay đổi mảng gốc.
- Các phương thức như push(), pop(), shift(), unshift() thay đổi mảng gốc.
- Luôn kiểm tra mảng có tồn tại và có độ dài trước khi truy cập: `arr && arr.length > 0`

10. Kết luận

Mảng là một cấu trúc dữ liệu cơ bản và quan trọng trong JavaScript. Việc nắm vững các phương thức và cách sử dụng mảng sẽ giúp bạn viết code hiệu quả và dễ đọc hơn. Hãy thực hành thường xuyên với các ví

dụ trên để củng cố kiến thức.