Funções Essenciais para Manipulação de Arrays

Os arrays em JavaScript vêm com uma variedade de métodos integrados que facilitam a manipulação de dados. Esta tabela resume as funções mais comuns e como usá-las.

Função	Descrição	Exemplo

push()	Adiciona um ou mais elementos ao **final** do array.	const frutas = ['maçã', 'banana'];
		frutas.push('morango');
		// frutas: ['maçã', 'banana', 'morango']

pop()	Remove o **último** elemento do array e o retorna.	const frutas = ['maçã', 'banana', 'morango'];
		const ultimo = frutas.pop();
		// frutas: ['maçã', 'banana']
		// ultimo: 'morango'

unshift()	Adiciona um ou mais elementos ao **início** do array.	const frutas = ['maçã', 'banana'];
		frutas.unshift('uva');
		// frutas: ['uva', 'maçã', 'banana']

shift()	Remove o **primeiro** elemento do array e o retorna.	const frutas = ['uva', 'maçã', 'banana'];
		const primeiro = frutas.shift();
		// frutas: ['maçã', 'banana']
		// primeiro: 'uva'

splice()	Um método versátil que **adiciona**, **remove** ou **substitui** elementos em qualquer posição.	const numeros = [1, 2, 3, 4, 5];
		numeros.splice(2, 1, 99);
		// Remove 1 elemento no índice 2 e adiciona 99
		// numeros: [1, 2, 99, 4, 5]

slice()	Retorna uma cópia rasa de uma porção do array, criando um **novo** array. Não modifica o original.	const cores = ['vermelho', 'verde', 'azul', 'amarelo'];
		const novasCores = cores.slice(1, 3);
		// novasCores: ['verde', 'azul']

indexOf()	Retorna o **primeiro índice** no qual um elemento pode ser encontrado no array. Retorna -1 se não o encontrar.	const nomes = ['Ana', 'João', 'Maria']; const indice = nomes.indexOf('João'); // indice: 1
includes()	Verifica se um array **inclui** um determinado elemento, retornando true ou false.	const nomes = ['Ana', 'João', 'Maria']; const existe = nomes.includes('João'); // existe: true

forEach()	Executa uma função para **cada elemento** do array. Não retorna um novo array.	const numeros = [1, 2, 3];
		numeros.forEach(num => console.log(num * 2));
		// Saída: 2, 4, 6

map()	Cria um **novo array** com os resultados da chamada de uma função para cada elemento do array.	const numeros = [1, 2, 3];
		const dobro = numeros.map(num => num * 2);
		// dobro: [2, 4, 6]

filter()	Cria um **novo array** com todos os elementos que passam em um teste implementado pela função fornecida.	const idades = [15, 20, 25, 30];
		const maioresDe20 = idades.filter(idade => idade > 20);
		// maioresDe20: [25, 30]

reduce()	Executa uma função redutora (fornecida pelo usuário) em cada elemento do array, resultando em um **único valor de retorno**.	const numeros = [1, 2, 3, 4];
		const soma = numeros.reduce((acumulador, num) => acumulador + num, 0);
		// soma: 10