

Aulas Java Script

Tópicos – Array (26/12/2025)

Um array é uma estrutura que guarda diversos valores em uma única variável.

Índices – arrays são indexados, e no JavaScript, se começa a contar no zero.

Tudo misturado – O array aceita diferentes tipos de variáveis no mesmo valor. Como números, strings ou booleanos.

Length – o código length é uma propriedade que mostra o tamanho do array (nome.length)

Como mudar um valor?

```
let cores = ["azul", "verde", "vermelho"];
cores[1] = "amarelo";

console.log(cores); // ["azul", "amarelo", "vermelho"]
```

Métodos comuns:

```
let numeros = [1, 2, 3];

numeros.push(4);    // adiciona no final
numeros.pop();     // remove o último
numeros.unshift(0); // adiciona no início
numeros.shift();   // remove o primeiro
```

Método Splice

O código splice tem a função de adicionar e remover valores de um array.

js

Copiar código

```
array.splice(inicio, quantidade, item1, item2, ...)
```

- `inicio` → índice onde começa a bagunça
- `quantidade` → quantos elementos serão removidos
- `item1, item2...` → itens que você quer adicionar (opcional)

```
let cores = ["azul", "vermelho"];

cores.splice(1, 0, "verde", "amarelo");

console.log(cores);
// ["azul", "verde", "amarelo", "vermelho"]
```

Concatenando arrays

js

Copiar código

```
// ARRAYS BASE
let a = [1, 2];
let b = [3, 4];
let c = [5, 6];

// 1. concat() -> cria um novo array, não altera os originais
let concatResultado = a.concat(b, c);
console.log("concat:", concatResultado);
// [1, 2, 3, 4, 5, 6]

// 2. spread operator (...) -> moderno e flexível, também não altera os originais
let spreadResultado = [...a, ...b, ...c];
console.log("spread:", spreadResultado);
// [1, 2, 3, 4, 5, 6]

// spread com valores no meio
let spreadComValor = [...a, 99, ...b];
console.log("spread com valor:", spreadComValor);
// [1, 2, 99, 3, 4]

// 3. push(...array) -> altera o array original
let pushArray = [1, 2];
pushArray.push(...b);
console.log("push:", pushArray);
// [1, 2, 3, 4]

// 4. Exemplo errado (array dentro de array)
let errado = [a, b];
console.log("errado:", errado);
// [[1, 2], [3, 4]]
```

Filters

```

// ARRAY BASE

let numeros = [1, 2, 3, 4, 5, 6, 7, 8];

// 1. Filtrar números pares
let pares = numeros.filter(numero => numero % 2 === 0);
console.log("pares:", pares);
// [2, 4, 6, 8]

// 2. Filtrar números ímpares
let impares = numeros.filter(numero => numero % 2 !== 0);
console.log("impares:", impares);
// [1, 3, 5, 7]

```

```

// 4. Filtrar strings por tamanho
let nomes = ["Ana", "João", "Carlos", "Lu"];

let nomesLongos = nomes.filter(nome => nome.length > 3);
console.log("nomes longos:", nomesLongos);
// ["João", "Carlos"]

// 5. Filtrar objetos
let alunos = [
  { nome: "Ana", nota: 7 },
  { nome: "Bruno", nota: 5 },
  { nome: "Carlos", nota: 8 }
];

let aprovados = alunos.filter(aluno => aluno.nota >= 6);
console.log("aprovados:", aprovados);
// [{ nome: "Ana", nota: 7 }, { nome: "Carlos", nota: 8 }]

```

- filter **não altera** o array original.
- Ele sempre retorna **um novo array**.
- A função dentro do filter **precisa retornar true ou false**.
- Se retornar true, o elemento entra no novo array.
- Se retornar false, ele é ignorado.

Map

```

// ARRAY BASE
let numeros = [1, 2, 3, 4, 5];

// 1. Dobrar valores
let dobrados = numeros.map(numero => numero * 2);
console.log("dobrados:", dobrados);
// [2, 4, 6, 8, 10]

// 2. Converter números em strings
let comoTexto = numeros.map(n => "Número: " + n);
console.log("como texto:", comoTexto);
// ["Número: 1", "Número: 2", "Número: 3", "Número: 4", "Número: 5"]

// 3. Calcular quadrado
let quadrados = numeros.map(n => n * n);
console.log("quadrados:", quadrados);
// [1, 4, 9, 16, 25]

// 4. Trabalhando com objetos
let alunos = [
  { nome: "Ana", nota: 7 },
  { nome: "Bruno", nota: 5 },
  { nome: "Carlos", nota: 8 }
];

// Extrair apenas os nomes
let nomes = alunos.map(aluno => aluno.nome);
console.log("nomes:", nomes);
// ["Ana", "Bruno", "Carlos"]

// 5. Criar novos objetos (sem alterar os originais)
let alunosComStatus = alunos.map(aluno => {
  return {
    nome: aluno.nome,
    nota: aluno.nota,
    status: aluno.nota >= 6 ? "Aprovado" : "Reprovado"
  };
});

console.log("alunos com status:", alunosComStatus);

```

- map **transforma**
 - Sempre retorna um **novo array**
 - O tamanho do array **não muda**
 - Cada return vira um elemento no novo array
 - Não use map para filtrar. Isso é pedir bug.
-

Reduce

Estrutura básica

js

 Copiar código

```
array.reduce((acumulador, valorAtual) => {  
  return novoValorDoAcumulador;  
}, valorInicial);
```

- **acumulador** → guarda o resultado parcial
- **valorAtual** → elemento atual do array
- **valorInicial** → ponto de partida do acumulador

```
// ARRAY BASE  
  
let numeros = [1, 2, 3, 4, 5];  
  
// 1. Somar todos os valores  
let soma = numeros.reduce((acumulador, valorAtual) => {  
  return acumulador + valorAtual;  
}, 0);  
  
console.log("soma:", soma);  
// 15  
  
// 2. Encontrar o maior número  
let maior = numeros.reduce((acc, atual) => {  
  return atual > acc ? atual : acc;  
}, numeros[0]);  
  
console.log("maior:", maior);  
// 5  
  
// 3. Multiplicar todos os valores  
let produto = numeros.reduce((acc, atual) => {  
  return acc * atual;  
}, 1);  
  
console.log("produto:", produto);  
// 120
```

```
// 4. Contar ocorrências
let letras = ["a", "b", "a", "c", "b", "a"];

let contagem = letras.reduce((acc, letra) => {
  acc[letra] = (acc[letra] || 0) + 1;
  return acc;
}, {});

console.log("contagem:", contagem);
// { a: 3, b: 2, c: 1 }

// 5. Somar notas de objetos
let alunos = [
  { nome: "Ana", nota: 7 },
  { nome: "Bruno", nota: 5 },
  { nome: "Carlos", nota: 8 }
];

let totalNotas = alunos.reduce((acc, aluno) => {
  return acc + aluno.nota;
}, 0);

console.log("total de notas:", totalNotas);
// 20
```

forEach

O que é o `forEach`?

É uma forma moderna de percorrer arrays quando você só quer executar código, não criar um novo array e não calcular um resultado final.

Pense assim:

`forEach` = "para cada item, faça isso".

Estrutura básica

js

 Copiar código

```
array.forEach((elemento, indice, arrayCompleto) => {  
    // código a ser executado  
});
```

Na prática, você quase sempre usa só o primeiro parâmetro.

● Nível 1 – O básico que você precisa saber (1–5)

Se errar aqui, o problema não é o array.

1. O que é um array em JavaScript?

O Array é um componente que pode armazenar diversos valores em uma única variável.

2. Para que tipos de dados um array pode armazenar valores?

O array pode armazenar qualquer tipo de dado disponibilizado pelo JavaScript. Além disso, um array permite guardar valores diferentes em uma variável, como números, strings e booleanos.

3. Qual a diferença entre um array e uma variável comum?

O array permite guardar mais de um valor.

4. O que significa dizer que arrays em JavaScript são indexados?

Indexados significa que as arrays tem posições são enumeradas. Começando pelo 0.

5. Por que o índice de um array começa em 0 e não em 1?

Para sistemas computacionais, começar a enumeração pelo 0 facilita toda a lógica.

● Nível 2 – Acesso e manipulação mental (6–10)

Aqui começa a exigir atenção, não força bruta.

6. O que acontece se você tentar acessar uma posição que não existe em um array?

O terminal retorna `undefined`.

7. Qual a diferença conceitual entre o **tamanho do array** e o **último índice**?

Quando falamos de tamanho de array, não estamos falando indexados. Quando queremos saber o tamanho do array usamos o `length`, e na contagem do tamanho, se inicia com o 1. Já o último índice, estamos falando de indexados, ai a contagem começa com 0.

8. O que significa dizer que arrays são mutáveis?

Significa que podemos adicionar, retirar ou alterar valores sem que haja um erro no sistema.

9. Qual a diferença entre adicionar um elemento no início e no fim de um array?

push - Quando adicionamos um elemento no fim do array, a lógica dos índices não muda.
unshift - Quando adicionamos no inicio, mudamos todos os índices do array, tendo grandes chances de haver erros de lógica no sistema.

10. Por que remover elementos de um array pode ser perigoso se você não souber o que está fazendo?

Pode ocorrer o mesmo erro da pergunta anterior. Aonde remover um elemento altera o índice da array.

● Nível 3 – Percorrendo arrays (11–15)

Aqui mora boa parte dos bugs do mundo.

11. O que significa “percorrer” ou “iterar” um array?

Percorrer significa que o programa passa pelos elementos do array.

Iterar - significa que o programa passa pelos elementos do array e executa uma função.

12. Qual a diferença conceitual entre percorrer um array com for e com forEach?

A diferença principal é que com o for o usuário pode controlar o loop.

Já o forEach, o loop é imutável.

13. Em que situação percorrer um array inteiro é desnecessário?

Para buscar elementos;

Verificar itens validos;

Validar condições.

14. Por que alterar um array enquanto você o percorre pode causar problemas?

Porque pode alterar a ordem do índice.

15. O que significa dizer que a ordem dos elementos em um array importa?

A ordem importa porque cada array tem seu significado.
Trocar os elementos muda o que o array está representando.

● Nível 4 – Pensamento funcional e armadilhas (16–20)

Aqui se separa quem entende de quem só usa.

16. Qual a diferença conceitual entre map, filter e reduce?

Map - Irá transformar cada elemento e gerar um novo array;
Filter - irá filtrar somente o que o usuário pedir;
Reduce - Junta o array em um único resultado.

17. Por que filter sempre retorna um novo array?

Porque esta opção não alterar o array original.

18. O que torna o reduce mais difícil de entender do que os outros métodos?

Pois ele é o método mais flexível de se manejar

19. Qual a diferença entre modificar o array original e retornar um novo array?

f

20. Em termos de boas práticas, quando **não** é uma boa ideia usar arrays?

Quando precisamos realizar buscar por palavras-chaves não por posição