ARRAYS
(AULA 8)

CURSO BÁSICO DE PROGRAMAÇÃO COM JAVASCRIPT

MAYARA MARQUES

mmrosatab@gmail.com

SUMÁRIO

- Operações em arrays
  - ForEach
  - Push
  - Filter
  - Map
  - Reduce
  - Some
  - Slice
  - Every
  - Includes
  - Spread Operator
  - Destructuring
- Mão na massa

FOREACH

1. **forEach**: Itera sobre cada elemento de um array e executa uma função para cada elemento.

   Exemplo:

```
const numbers = [1, 2, 3]
numbers.forEach(number => console.log(number))
console.log(numbers) // 1 2 3
```

PUSH

2. **push**: Adiciona um elemento ao final de um array.

Exemplo:

```
const fruits = ['apple', 'banana']
fruits.push('orange')
console.log(fruits) // ['apple', 'banana', 'orange']
```

FILTER

3. **filter**: Cria um novo array com elementos que atendem a um critério de filtro.

Exemplo:

```
const numbers = [1, 2, 3, 4, 5]
const evenNumbers = numbers.filter(number => number % 2 === 0)
console.log(evenNumbers) // [2, 4]
```

MAP

4. **map**: Cria um novo array aplicando uma função a cada elemento do array original.

Exemplo:

```
const numbers = [1, 2, 3]
const doubled = numbers.map(number => number * 2)
console.log(doubled) // [2, 4, 6]
```

REDUCE

5. **reduce**: Reduz um array a um único valor aplicando uma função acumuladora.

Exemplo:

```
const numbers = [1, 2, 3, 4]
const sum = numbers.reduce((accumulator, current) => accumulator + current, 0)
console.log(sum) // 10
```

SOME

6. **some**: Verifica se pelo menos um elemento atende a um critério e retorna true ou false.

Exemplo:

```
const numbers = [1, 2, 3]
const hasEven = numbers.some(number => number % 2 === 0)
console.log(hasEven) // true
```

SLICE

7. **slice**: Cria uma cópia de um array com base nos índices especificados.

Exemplo:

```
const fruits = ['apple', 'banana', 'cherry']
const subset = fruits.slice(1, 3)
console.log(subset) // ['banana', cherry]
```

EVERY

8. **every**: Verifica se todos os elementos atendem a um critério e retorna true ou false.

Exemplo:

```
const numbers = [2, 4, 6]
const allEven = numbers.every(number => number % 2 === 0)
console.log(allEven) // true
```

INCLUDES

9. **includes**: Verifica se um elemento específico está presente no array e retorna true ou false.

Exemplo:

```
const fruits = ['apple', 'banana', 'cherry']
const hasBanana = fruits.includes('banana')
console.log(hasBanana) // true
```

SPREAD OPERATOR

10. **spread operator**: Espalha os elementos de um array ou objeto em outro array ou objeto. Ele é muito útil para **copiar**, **combinar** e **modificar** arrays **sem** alterar os originais.

Exemplo:

```
const arr1 = [1, 2, 3]
const arr2 = [...arr1, 4, 5]
// [1, 2, 3, 4, 5]
```

DESTRUCTING

11. **destructuring**: Permite extrair valores de arrays ou objetos em variáveis individuais.

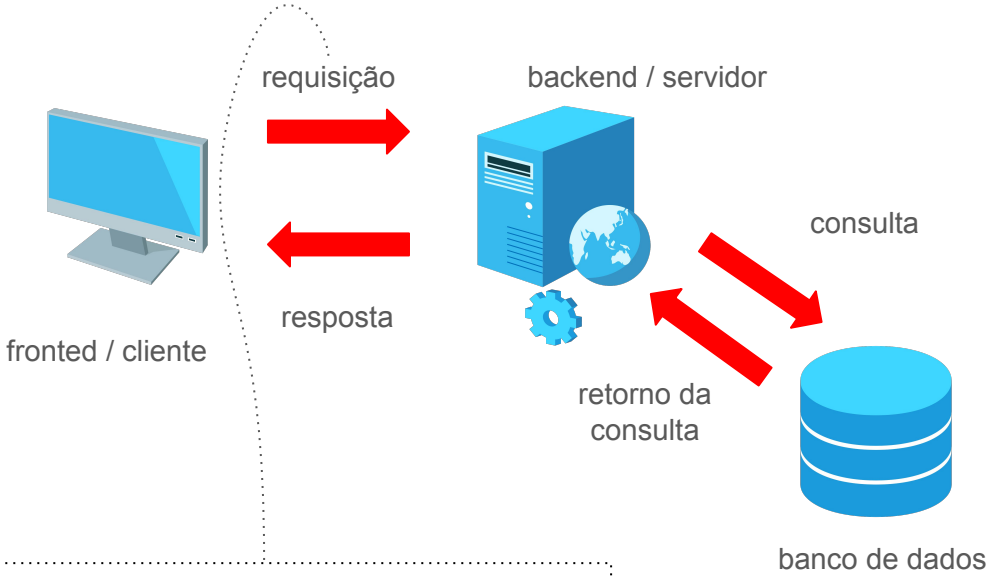
Exemplo:

```
const numbers = [1, 2, 3]

// Destructuring para extrair valores individuais
const [first, second, third] = numbers

console.log(first)  // 1
console.log(second) // 2
console.log(third)  // 3
```

MÃO NA MASSA

MÃO NA MASSA

1. Imagine que a variável **response** é a resposta de uma requisição feita para o backend contendo os dados de todos os personagens cadastrados no banco de dados de uma aplicação web. Como você faria para recuperar o personagem de id=3 ? (OBS: copie o array **response** abaixo para utilizar em seu programa)



Uma **requisição** é um pedido/solicitação de uma informação ou ação feito de um sistema para outro.

```
const response = [
  {
    id: '1',
    name: 'Yennefer',
    blood: 'Half-elven',
    classification: 'Witch',
  },
  {
    id: '2',
    name: 'Geralt',
    blood: 'Human/Mutant',
    classification: 'Wizard',
  },
  {
    id: '3',
    name: 'Ciri',
    blood: 'Half-elven',
    classification: 'Witch',
  },
  {
    id: '4',
    name: 'Jaskier',
    blood: 'Human',
    classification: 'Poetry',
  },
]
```

MÃO NA MASSA

2. Como você faria para recuperar os personagens de sangue meio-elfo (Half-elven)? (OBS: copie o array abaixo para utilizar em seu programa)

```
const response = [
    {
        id: '1',
        name: 'Yennefer',
        blood: 'Half-elven',
        classification: 'Witch',
    },
    {
        id: '2',
        name: 'Geralt',
        blood: 'Human/Mutant',
        classification: 'Wizard',
    },
    {
        id: '3',
        name: 'Ciri',
        blood: 'Half-elven',
        classification: 'Witch',
    },
    {
        id: '4',
        name: 'Jaskier',
        blood: 'Human',
        classification: 'Poetry',
    },
]
```

MÃO NA MASSA

3. Imagine que os dados de response, tanto chave quanto valor, serão utilizados em outro ponto da aplicação. Contudo, foi determinado que as propriedades **name**, **blood** e **classification** devem ser alteradas para **nome**, **sangue** e **classificacao**, e além disso, valor para as chaves **blood** e **classification** devem ser traduzidos para o português . Como você faria para criar a variável newFormat a partir dos dados que estão em response ? (OBS: copie o array abaixo para utilizar em seu programa)

```
const response = [
  {
    id: '1',
    name: 'Yennefer',
    blood: 'Half-elven',
    classification: 'Witch',
  },
  {
    id: '2',
    name: 'Geralt',
    blood: 'Human/Mutant',
    classification: 'Wizard',
  },
  {
    id: '3',
    name: 'Ciri',
    blood: 'Half-elven',
    classification: 'Witch',
  },
  {
    id: '4',
    name: 'Jaskier',
    blood: 'Human',
    classification: 'Poetry',
  },
]
```

```
const newFormat = [
  {
    id: '1',
    nome: 'Yennefer',
    sangue: 'Meio-elfo',
    classificacao: Bruxa,
  },
  {
    id: '2',
    nome: 'Geralt',
    sangue: 'Humano/Mutante'
    classificacao: Bruxo,
  },
  {
    id: '3',
    name: 'Ciri',
    sangue: 'Meio-elfo',
    classificacao: Bruxa,
  },
  {
    id: '4',
    name: 'Jaskier',
    sangue: 'Humano',
    classificacao: Poeta,
  },
]
```

MÃO NA MASSA

4. Crie um programa que receba um array de números inteiros e retorne o produtório desses valores.

5. Crie um programa que receba um array de nomes e um nome de pesquisa e retorne true ou false para dizer se o nome específico está presente no array.

6. Baseado no programa anterior, ache o índice do array que contém o nome de pesquisa.

7. Crie um programa que receba um array de números inteiros e retorne true caso todos os números sejam primos e falso caso contrário.

DESAFIO

Como programadores, alguns dos nossos desafios são pesquisar, entender e aplicar novos conceitos quase que diariamente. Para nos acostumarmos com esse ritmo, vamos fazer uma pesquisa antes de iniciar o desafio.

Pesquise e estude a função **sort** utilizada em arrays no JavaScript**.** Entenda seu funcionamento tanto em arrays de números quanto arrays de strings.

Após a pesquisa e entendimento, você deve elaborar um programa que resolva o problema elucidado no próximo slide.

DESAFIO

O Mercado Inteligente 🛒

Você foi contratado para desenvolver um sistema para um **mercado inteligente**. O objetivo é ajudar os funcionários a organizarem os produtos de forma eficiente para melhorar a experiência dos clientes.

O gerente do mercado quer uma **ferramenta automatizada** para classificar os produtos de acordo com diferentes critérios, pois isso facilitará a reposição nas prateleiras e melhorará a experiência de compra.

Seu desafio é criar um programa que ordene os produtos de **forma dinâmica**, dependendo da necessidade do mercado.

DESAFIO

🛒 O Problema

O mercado possui uma **lista de produtos** que inclui:

- Nome do produto
- Categoria (ex: "alimentos", "bebidas", "higiene")
- Preço
- Popularidade (quantidade de vendas no mês)

Os funcionários querem **ordenar essa lista** de diferentes formas:

1. **Por nome do produto** (ordem alfabética, ignorando maiúsculas e minúsculas).
2. **Por categoria**, para organizar as prateleiras (ex: alimentos primeiro, depois bebidas e higiene).
3. **Por preço**, do menor para o maior.
4. **Por popularidade**, do mais vendido para o menos vendido.

Você deve criar uma **função** que permita ordenar os produtos conforme a necessidade do momento.

DESAFIO

O que deve ser feito?

- ★ Criar um array de produtos com nome, categoria, preço e popularidade.
- ★ Implementar uma **função** que receba um critério de ordenação e organize os produtos dinamicamente.
- ★ Exibir a lista antes e depois da ordenação.

O que deve ser apresentado?

- ★ Qual a estrutura da função sort?
  - ○ Quais parâmetros pode receber?
  - ○ A função tem algum valor de retorno ou não?
  - ○ A função modifica o array passado ou cria um novo array ordenado?
- ★ Como a função sort funciona para strings e números?
- ★ Explicar como resolveu o desafio passo a passo.

Prazo: uma semana!!!

DESAFIO

Exemplo de entrada e saída

```
const produtos = [
    { nome: "Arroz", categoria: "alimentos", preco: 20, popularidade: 500 },
    { nome: "Sabonete", categoria: "higiene", preco: 5, popularidade: 300 },
    { nome: "Refrigerante", categoria: "bebidas", preco: 8, popularidade: 800 },
    { nome: "Feijão", categoria: "alimentos", preco: 12, popularidade: 600 },
    { nome: "Cerveja", categoria: "bebidas", preco: 10, popularidade: 1000 }
]

const produtosOrdenados = ordenarProdutos(produtos, "nome")

console.log(produtosOrdenados)

// saída
[
    { nome: "Arroz", categoria: "alimentos", preco: 20, popularidade: 500 },
    { nome: "Cerveja", categoria: "bebidas", preco: 10, popularidade: 1000 },
    { nome: "Feijão", categoria: "alimentos", preco: 12, popularidade: 600 },
    { nome: "Refrigerante", categoria: "bebidas", preco: 8, popularidade: 800 },
    { nome: "Sabonete", categoria: "higiene", preco: 5, popularidade: 300 }
]
```

DESAFIO

Exemplo de entrada e saída

```
const produtos = [
    { nome: "Arroz", categoria: "alimentos", preco: 20, popularidade: 500 },
    { nome: "Sabonete", categoria: "higiene", preco: 5, popularidade: 300 },
    { nome: "Refrigerante", categoria: "bebidas", preco: 8, popularidade: 800 },
    { nome: "Feijão", categoria: "alimentos", preco: 12, popularidade: 600 },
    { nome: "Cerveja", categoria: "bebidas", preco: 10, popularidade: 1000 }
]

const produtosOrdenados = ordenarProdutos(produtos, "preco")

console.log(produtosOrdenados)

// saída
[
    { nome: "Sabonete", categoria: "higiene", preco: 5, popularidade: 300 },
    { nome: "Refrigerante", categoria: "bebidas", preco: 8, popularidade: 800 },
    { nome: "Cerveja", categoria: "bebidas", preco: 10, popularidade: 1000 },
    { nome: "Feijão", categoria: "alimentos", preco: 12, popularidade: 600 },
    { nome: "Arroz", categoria: "alimentos", preco: 20, popularidade: 500 }
]
```

REFERÊNCIAS

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array
- https://www.w3schools.com/jsref/jsref_obj_array.asp