

Algoritmia e Programação

**Estruturas de Dados
Arrays, Objetos e Coleções**

Objetivos

- Compreender o que é uma estrutura de dados
- Utilizar arrays em JavaScript de forma correta e consciente
- Aplicar operações fundamentais sobre arrays
- Implementar algoritmos simples de ordenação e pesquisa
- Criar objetos e classes
- Trabalhar com arrays de objetos
- Reconhecer outras estruturas de dados e os seus contextos de uso

ARRAYS - O que é um Array?

- Um **array** é:
 - Uma estrutura de dados linear
 - Que armazena vários valores
 - Indexados a partir de 0
 - Um array em JavaScript é um objeto especial

```
const numeros = [5, 2, 9, 1];
```

```
const numeros = [];
```

```
numeros[0] = 5;
```

```
numeros[1] = 2;
```

```
numeros[2] = 9;
```

```
numeros[3] = 1;
```

ARRAYS - Particularidades dos Arrays em JavaScript

- Os **arrays** em JavaScript:
 - São dinâmicos (tamanho variável)
 - Podem conter tipos mistos
 - São objetos
 - São passados nas funções por referência
 - Encapsulam muitos métodos

```
const a = [1, "texto", true];
```

ARRAYS - Índices e Acesso

- Índices inválidos devolvem **undefined**
- Não gera erro

```
const numeros = [10, 20, 30];

console.log(numeros[0]); // 10
console.log(numeros[2]); // 30
```

ARRAYS - Tamanho do Array

- **length:**
 - Não é o último índice
 - Pode ser alterado (não recomendado)

```
numeros.length = 1; // corta o array
```

ARRAYS - Percorrer um Array

```
for (let i = 0; i < numeros.length; i++) {  
    console.log(numeros[i]);  
}
```

```
for (let n in numeros){  
    console.log(numeros[n]);  
}
```

```
for (let n of numeros) {  
    console.log(n);  
}
```

ARRAYS - Exercício 1

- Criar um array de inteiros e calcular a soma dos elementos.

```
const numeros = [2, 4, 6, 8];
let soma = 0;

for (let i = 0; i < numeros.length; i++) {
    soma += numeros[i];
}

console.log(soma);
```

ARRAYS - Exercício 2

- Criar um programa que:
 - Guarde 10 números inteiros num array
- Calcule:
 - maior valor
 - menor valor
 - média

ARRAYS - Pesquisa de Elementos

- Pesquisar significa:
 - Verificar se um valor existe
 - Descobrir a sua posição
 - Exemplo - Pesquisa Sequencial (Linear)
- JavaScript oferece métodos,
mas vamos implementar o algoritmo

```
function pesquisaLinear(arr, valor) {  
    let i = 0;  
  
    while (i < arr.length && arr[i] != valor) {  
        i++;  
    }  
  
    if (i >= arr.length) {  
        return -1  
    }  
  
    return 1;  
}
```

ARRAYS - Exercício 3

- **Modificar a pesquisa linear para:**
 - Contar quantas vezes um valor aparece
 - Não parar na primeira ocorrência

ARRAYS - ORDENAÇÃO

- Ordenar significa:
 - Reorganizar os dados segundo um critério
 - Ascendente ou descendente
- Ordenação por Seleção
 - Ideia:
 - Seleciona o menor/maior
 - Coloca-o na posição correta
 - Compara arr[i] com todos os restantes

```
function selectionSort(arr) {  
    for (let i = 0; i < arr.length - 1; i++) {  
        for (let j = i + 1; j < arr.length; j++) {  
            if (arr[j] > arr[i]) {  
                let temp = arr[i];  
                arr[i] = arr[j];  
                arr[j] = temp;  
            }  
        }  
    }  
}
```

ARRAYS - Métodos Essenciais

- **Métodos mais importantes:**

- push, pop
- shift, unshift
- splice
- slice
- indexOf, includes

ARRAYS - Adicionar e Remover Elementos (métodos)

- **shift** e **unshift** são mais caros computacionalmente
- Removem / adicionam elementos no início do array.
- Características:
 - Alteram o array original
 - Todos os elementos são reposicionados
 - Custo elevado em arrays grandes
- **push** e **pop** adicionam e removem do fim respetivamente

```
const a = [];  
  
a.push(10);      // adiciona no fim  
a.pop();         // remove do fim  
  
a.unshift(5);   // adiciona no início  
a.shift();       // remove do início
```

Exercício rápido : Remover o primeiro elemento de um array apenas se o array não estiver vazio.

ARRAYS - Adicionar e Remover Elementos (métodos)

- `splice()`
- Permite:
 - remover
 - inserir
 - substituir elementos

```
let numeros = [10, 20, 30, 40];
numeros.splice(1, 2);
// remove 2 elementos a partir do índice 1
// resultado: [10, 40]

//inserir (0 no segundo argumento)
numeros.splice(1, 0, 15);
// [10, 15, 40]

//substituir (1 no segundo argumento)
numeros.splice(1, 1, 99);
// [10, 99, 40]
```

ARRAYS - Copiar array (métodos)

- **slice()**
 - Cria uma cópia parcial ou integral do array.

```
//copia integral
const original = [1, 2, 3];
const copia = original.slice();
```

```
const numeros = [10, 20, 30, 40];
const parte = numeros.slice(1, 3); //do indice 1 ao 3

console.log(parte);    // [20, 30]
console.log(numeros); // não é alterado
```

ARRAYS - Pesquisa (métodos)

- **Indexof e includes**
 - Pesquisa simples em arrays.

```
const numeros = [5, 10, 15];  
  
numeros.indexOf(10); // 1  
numeros.indexOf(99); // -1  
  
numeros.includes(15); // true  
numeros.includes(99); // false
```

Exercício rápido : Verificar se um número existe no array sem usar includes().

ARRAYS - map() (métodos)

- Cria um novo array transformando cada elemento.
- Não altera o array original.
- Quando usar map
 - Transformação 1 → 1
 - Mesmo tamanho
 - Sem efeitos colaterais

```
const numeros = [1, 2, 3];
const quadrados = numeros.map((n) => n * n);

console.log(quadrados); // [1, 4, 9]
```

ARRAYS - reduce() (métodos)

- Reduz o array a um único valor.
- ?
 - acc → acumulador
 - 0 → valor inicial

```
const numeros = [1, 2, 3, 4];

const soma = numeros.reduce((acc, n) => acc + n, 0);

console.log(soma); // 10
```

ARRAYS - reduce() (métodos)

- Reduz o array a um único valor.
- ?
 - acc → acumulador
 - 0 → valor inicial
- Usos de reduce
 - soma
 - produto
 - máximo / mínimo
 - contagem
 - Agregações

```
const numeros = [1, 2, 3, 4];  
  
const soma = numeros.reduce((acc, n) => acc + n, 0);  
  
console.log(soma); // 10
```

ARRAYS - reduce() (métodos)

- Calcular o maior valor de um array.

```
const numeros = [4, 7, 2, 9];

const max = numeros.reduce((m, n) => (n > m ? n : m));

console.log(max);
```

Algoritmia e Programação

**Estruturas de Dados
Arrays, Objetos e Coleções**