

Estrutura de Dados

MSc Sandir Rodrigues Campos

Estrutura de dados - revisando

- Criação de array:
 - `var frutas = ['Maçã', 'Banana'];`
 - Acessar primeiro item (index) do Array
 - `var ultimo = frutas[0];`
 - Acessar último item (index) do Array
 - `var ultimo = frutas[frutas.length - 1];`
 - Acessar item X(index) do Array
 - `var ultimo = frutas[X];`
-
- Iterar um Array
 - `frutas.forEach(function (item, indice, array) {`
 - `console.log(item, indice);`
 - `});`
 - Adicionar um item ao final do Array
 - `var adicionar = frutas.push('Laranja');`
 - Remover um item do final do Array
 - `var ultimo = frutas.pop();`
 - Remover do início do Array
 - `var primeiro = frutas.shift();`

Estrutura de Dados

Adicionando ao início do Array:

```
var adicionar = frutas.unshift('Morango') // adiciona ao  
início
```

```
// ['Morango', 'Banana'];
```

Estrutura de Dados

Métodos Importantes



Estrutura de Dados



método splice() - usado para adicionar ou remover elementos de um array.



Sintaxe:

`array.splice(index, quantidade, elemento1, elemento2, ...)`



Parâmetros:

index: O índice onde você deseja começar a modificar o array.

quantidade: O número de elementos que você deseja remover do array.

elemento1, elemento2, ...: Os elementos que você deseja adicionar ao array.

Estrutura de Dados - Método splice()

- **Remover elementos:**
 - `var frutas = ["maçã", "banana", "laranja","abacate"];`
 - `frutas.splice(1, 2);console.log(frutas);`
 - `// Saída: ["maçã", "abacate"]`
- **Adicionar elementos:**
 - `var frutas = ["maçã", "banana", "laranja"];`
 - `frutas.splice(1, 0, "manga", "pêra");`
 - `console.log(frutas);`
 - `// Saída: ["maçã", "manga", "pêra", "banana", "laranja"]`

Estrutura de Dados - Método splice()

- **Remover e adicionar elementos:**

- `var frutas = ["maçã", "banana", "laranja", "abacate"];`
- `frutas.splice(1, 2, "manga", "pêra");`
- `console.log(frutas);`
- `// Saída: ["maçã", "manga", "pêra", "abacate"]`

Observações:

- O método `splice()` modifica o array original.
- Se você não especificar o parâmetro quantidade, todos os elementos a partir do índice especificado serão removidos.
- Se você não especificar os parâmetros elemento1, elemento2, ..., nenhum elemento será adicionado.

Estrutura de Dados - Método sort()



método sort() - usado para ordenar os elementos de um array. Ele é muito útil quando você precisa ordenar dados de forma alfabética ou numérica.



Sintaxe:

```
array.sort([funçãoDeComparaç  
ão])
```



Parâmetros:

- funçãoDeComparação: Uma função que define a ordem dos elementos. Se não for fornecida, o array será ordenado de forma alfabética.

Estrutura de Dados - Método sort()

Ordenar um array de strings:

- `var frutas = ["maçã", "banana", "laranja", "abacate"];`
- `frutas.sort();`
- `console.log(frutas);`
- `// Saída: ["abacate", "banana", "laranja", "maçã"]`

Ordenar um array de números:

- `var números = [4, 2, 7, 1, 3];`
- `números.sort();`
- `console.log(números);`
- `// Saída: [1, 2, 3, 4, 7]`

Estrutura de Dados - Método sort()

- Ordenar um array de objetos:

```
var pessoas = [ {nome: "João",  
idade: 25}, {nome: "Maria", idade:
```

```
30}, {nome: "Pedro", idade: 20}];
```

```
pessoas.sort(function(a, b) {  
    return a.idade - b.idade;  
});
```

```
console.log(pessoas);
```

```
// Saída: [{nome: "Pedro", idade: 20},  
{nome: "João", idade: 25}, {nome:  
"Maria", idade: 30}]
```



Estrutura de Dados - Método sort()

- Função de comparação: A função de comparação é uma função que define a ordem dos elementos. Ela recebe dois parâmetros, **a** e **b**, que são os elementos a serem comparados. Se a função de comparação retornar:- Um valor negativo, **a** será colocado antes de **b**.- Um valor positivo, **a** será colocado após **b**.- Zero, a ordem de **a** e **b** não será alterada.
- Observações:
 - O método sort() modifica o array original.
 - Se você não fornecer uma função de comparação, o array será ordenado de forma alfabética.
 - A função de comparação pode ser uma função anônima ou uma função nomeada.

Estrutura de Dados


- Método concat()

método concat() - usado para concatenar (unir) dois ou mais arrays em um novo array.

Sintaxe: `array1.concat(array2, array3, ...)`


Parâmetros:

- array1, array2, array3, ...: Os arrays que você deseja concatenar.
- Return:
 - Um novo array que contém todos os elementos dos arrays concatenados.



Estrutura de Dados - Método concat()

- Concatenar dois arrays:

```
var frutas = ["maçã", "banana",  
"laranja"];  
var legumes = ["cenoura", "batata",  
"ervilha"];  
var alimentos =  
frutas.concat(legumes);  
console.log(alimentos);  
// Saída: ["maçã", "banana", "laranja",  
"cenoura", "batata", "ervilha"]
```
- 

Estrutura de Dados - Método concat()

- Concatenar três arrays:

```
var frutas = ["maçã", "banana", "laranja"];  
var legumes = ["cenoura", "batata", "ervilha"];  
var verduras = ["alface", "espinafre", "rúcula"];  
var alimentos = frutas.concat(legumes,  
verduras);  
console.log(alimentos);  
// Saída: ["maçã", "banana", "laranja", "cenoura",  
"batata", "ervilha", "alface", "espinafre",  
"rúcula"]
```

Estrutura de Dados - Método concat()

- Concatenar arrays com valores diferentes:

```
var números = [1, 2, 3];  
var letras = ["a", "b", "c"];  
var mistura = números.concat(letras);  
console.log(mistura);  
// Saída: [1, 2, 3, "a", "b", "c"]
```

Observações:- O método concat() não modifica os arrays originais.

- O método concat() pode ser usado para concatenar arrays com diferentes tipos de dados.
- O método concat() é uma forma mais segura de concatenar arrays do que usar o operador +, pois evita a conversão de tipos de dados.

Estrutura de Dados - Método startsWith()

método `startsWith()`: usado para verificar se uma string começa com uma determinada substring.

Sintaxe: `string.startsWith(substring, posição)`

Parâmetros:

`substring`: A substring que você deseja verificar se a string começa com ela.

`posição`: Opcional. A posição inicial da string onde você deseja começar a verificar. Se não for fornecida, a verificação começará na posição 0

Return:

`true` se a string começa com a substring, `false` caso contrário.

Estrutura de Dados - Método startsWith()

Verificar se uma string começa com "c":

- `var string = "casa";`
- `console.log(string.startsWith("c")); // Saída: true`

Verificar se uma string começa com "b" a partir da posição 2:

- `var string = "abacate";`
- `console.log(string.startsWith("b", 1)); // Saída: true`

Verificar se uma string não começa com "c":

- `var string = "amigo";`
- `console.log(string.startsWith("c")); // Saída: false`

Estrutura de Dados - Método startsWith()

Observações:

- O método `startsWith()` é sensível a maiúsculas e minúsculas.
- O método `startsWith()` pode ser usado com strings que contenham caracteres especiais.
- O método `startsWith()` é uma forma eficiente de verificar se uma string começa com uma determinada substring.

Também é possível usar o método `startsWith()` com expressões regulares:

```
var string = "amigo";  
console.log(string.startsWith(/a/)); // Saída: true
```

Estrutura de Dados - Método filter ()

método filter(): usado para criar um novo array com todos os elementos que atendem a uma condição específica.

Sintaxe:array.filter(funçãoDeTeste)

Parâmetros:

- funçãoDeTeste: Uma função que define a condição para incluir um elemento no novo array.

Return:

- Um novo array com todos os elementos que atendem à condição definida pela funçãoDeTeste.

Estrutura de Dados

- Método filter()

- Filtrar números pares:

```
var números = [1, 2, 3, 4, 5, 6];  
var númerosPares =  
  números.filter(function(num) {  
    return num % 2 === 0;  
  });  
console.log(númerosPares);  
// Saída: [2, 4, 6]
```

Estrutura de Dados - Método filter()

- Filtrar strings que começam com "a":

```
var palavras = ["amigo", "banana", "abacate", "laranja"];  
var palavrasA = palavras.filter(function(palavra) {  
    return palavra.startsWith("a");  
});  
console.log(palavrasA);  
// Saída: ["amigo", "abacate"]
```

Estrutura de Dados - Método filter()

- • Filtrar objetos com propriedade específica:
var pessoas = [{nome: "João", idade: 25}, {nome: "Maria", idade: 30},
{nome: "Pedro", idade: 20}];
var pessoasMaiores = pessoas.filter(function(pessoa) {

 return pessoa.idade > 25;

});
console.log(pessoasMaiores); // Saída: [{nome: "Maria", idade: 30}]

Estrutura de Dados - Método filter()

Observações:

- O método filter() não modifica o array original.

- O método filter() pode ser usado com arrays de diferentes tipos de dados.

- O método filter() é uma forma eficiente de filtrar dados em arrays.

Também é possível usar **arrow functions** para simplificar a sintaxe:

```
var númerosPares = números.filter(num => num % 2 === 0);
```