**Método splice()**

Utilize o **método javascript splice()** quando você precisar remover itens de um array enquanto adiciona um ou mais.

Este método recebe um indíce como primeiro argumento, onde especifica a posição que deve iniciar a remoção, o segundo argumento a quantidade de itens para remover e o terceiro em diante, são os novos elementos para adicionar. O retorno deste método, são os itens removidos.

const fruits = ['Apple', 'Orange', 'Banana'];

console.log(fruits); // Output: ["Apple", "Orange", "Banana"]

const removed = fruits.splice(1, 1, 'Papaya');

console.log(removed); // Output: ["Orange"]

console.log(fruits); // Output: ["Apple", "Papaya", "Banana"]

**Método indexOf()**

Utilize o **método javascript indexOf()** quando você precisa descobrir a posição de um elemento no array. Quando não for encontrado, o resultado será -1.

const fruits = ['Apple', 'Orange', 'Banana'];

console.log(fruits.indexOf('Orange')); // Output: 1

console.log(fruits.indexOf('Papaya')); // Output: -1

**Método lastIndexOf()**

Utilize o **método javascript lastIndexOf()** para retornar a última posição de um item no array. A pesquisa no array é realizada de trás para frente.

Opcionalmente, você pode definir uma posição inicial para a busca de traz para frente. Quando nenhuma ocorrência é encontrada para o item, o resultado será -1.

const fruits = ['Apple', 'Orange', 'Banana', 'Apple'];

console.log(fruits.lastIndexOf('Apple')); // Output: 3

console.log(fruits.lastIndexOf('Apple', 1)); // Output: 0

console.log(fruits.lastIndexOf('Papaya')); // Output: -1

**Método find()**

Utilize o **método javascript find()** se você precisar retornar o valor de um elemento do array. O método find executa a função *callback* testando cada elemento do array, até encontrar o valor desejado e o *callback* retornar true.

// First Sample

let fruits = [ 3, 4, 7, 9 ];

console.log(fruits.find(element => element > 5)); // Output: 7

// Second Sampel

fruits = [

{ name: 'Apple', price: 3.55 },

{ name: 'Orange', price: 2.43 },

{ name: 'Papaya', price: 4.76 }

];

const result = fruits.find(fruit => fruit.name === 'Apple');

console.log(result); // Output: { name: "Apple", price: 3.55 }

**Método reverse()**

Utilize o **método javascript reverse()** se você precisar inverter os itens de um array, o último item vira o primeiro e o primeiro o último.

const list = [1, 2, 3, 4];

console.log(list.reverse()); // Output: [4,3,2,1]

**Método sort()**

Utilize o **método javascript sort()** se você precisar classificar elementos de um array. Este método retorna um array ordenado e altera as posições do array original.

Por padrão, a classificação é em ordem alfabética, veja o seguinte exemplo:

const fruits = ['orange', 'pineapple', 'apple', 'papaya'];

fruits.sort()

console.log(fruits); // Output: ["apple", "orange", "papaya", "pineapple"]

Para a classificação ser em ordem alfabética decrescente, você precisa fazer uma pequena mudança na função. Veja o exemplo:

const fruits = ['orange', 'pineapple', 'apple', 'papaya'];

fruits.sort((a, b) => {

if (a > b) return -1;

if (a < b) return 1;

return 0

});

console.log(fruits);

Estes são os dois exemplos básicos, usando a função de comparação padrão. A função padrão baseia-se nos valores dos pontos de código Unicode dos elementos.

Se você precisar, pode mudar a função de comparação, respeitando as seguintes regras:

1. Se myCustomCompare(a, b) for **menor que zero**, ordena “**a**” para o índice anterior a “**b**“.

2. Se myCustomCompare(a, b) for **maior que zero**, ordena “**b**” para o índice anterior ao “**a**“.

3. Se myCustomCompare(a, b) retornar zero, o “a” e “b” ficam na posição.

const numbers = [0, 1, 10, 2, 3, 15, 20];

numbers.sort();

console.log(numbers); // Output: [0, 1, 10, 15, 2, 20, 3]

numbers.sort((a, b) => {

if (a > b) return 1;

if (a < b) return -1;

return 0

});

console.log(numbers); // Output: [0, 1, 2, 3, 10, 15, 20]

**Método** **shift()**

Utilize o **método javascript shift()** para remover e retornar o primeiro elemento de um array. Quando executado, o array original muda.

const fruits = ['Apple', 'Papaya', 'Orange'];

const firstFruit = fruits.shift();

console.log(firstFruit); // Output: Apple

console.log(fruits); // Output: [ 'Papaya', 'Orange' ]

**Método** **pop()**

Utilize o **método javascript pop()** para remover e retornar o último elemento de um array. Quando executado, o array original muda.

const fruits = ['Apple', 'Papaya', 'Orange'];

const firstFruit = fruits.pop();

console.log(firstFruit); // Output: Orange

console.log(fruits); // Output: [ 'Apple', 'Papaya' ]

**Método push()**

Utilize o **método javascript push()** para adicionar um ou mais elementos no final de um array e retornar o novo tamanho deste array.

const languages = ['javascript', 'golang', 'nodejs'];

const total = languages.push('c#', 'elixir');

console.log(languages); // Output: ["javascript", "golang", "nodejs", "c#", "elixir"]

console.log(total); // Output: 5

**Método unshift()**

Utilize o **método javascript unshift()** para adicionar um ou mais elementos no início de um array e retornar o novo tamanho deste array.

const total = languages.unshift('c#', 'elixir');

console.log(languages); // Output: ["c#", "elixir", "javascript", "golang", "nodejs"]

console.log(total); // Output: 5