Fundamentos do JavaScript: Arrays

PARTE 1

Introdução à Arrays:

Array: Um tipo de estrutura de dados em JS.

Pode ser utilizada, por exemplo, para agrupar diversos dados que tem relação entre si:

```
// sem array:

const numero1 = 50;
const numero2 = 43;
const numero3 = 12;

// utilizando array
const numeros = [50, 43, 12];
```

Definindo array -> Lista ORDENADA de valores enumerados em que cada valor é chamado de ELEMENTO, e cada elemento se localiza em uma posição na lista chamada de índice.

Em programação, chamamos de índice (em inglês, ***index***) o número que identifica a posição de um valor em um array.

A contagem em arrays na linguagem JS começa no índex [0]

Propriedade 'length' significa 'tamanho', e ela faz a função de contar sozinha o tamanho do array (quantos itens tem nesse array), nós não precisamos realizar esse trabalho.

```
// índice          0  1   2   3
const nums = [50, 43, 12, 98];
console.log(nums.length) //4
```

Adicionando elementos em um Array:

Método

.push (adiciona um novo item ao FINAL do array)

```
const notas = [10, 6, 8]
notas.push(7) //adicionando ao final do array
console.log(notas)
```

adicionou para mim o número '7' ao final do array notas [10, 6, 8, 7]

Deletando elementos em um Array:

Método

.pop (tira o último elemento do array)

```
const notas = [10, 6, 8, 5.5, 10]
notas.pop()
console.log(notas)
```

o método (.pop) não recebe nenhum parâmetro, porque o JS já interpreta que você quer tirar o último elemento do array.

E seguindo o desafio e calculando a média:

```
let media = (notas[0] + notas[1] + notas[2] + notas[3]) / notas.length
console.log(`A média é ${media}`)
// recebi como retorno: 7.375
```

Utilizei também o modo de escrita “templateString” => ${ }, que é uma forma mais simples e não precisa concatenar com “+”

Principais métodos de manipulação de Arrays:

concat()

Junta dois arrays, colocando o array passado como argumento, logo depois do primeiro. Em português essa operação é conhecida como concatenação.

Não altera o array no qual foi chamado, então precisamos salvar esse resultado em um novo array.

filter()

Retorna uma lista contando todos os elementos que passaram em um teste, ou seja, uma função escrita por nós.

Não altera o array onde foi chamado, então precisamos salvar esse resultado em um novo array.

find()

Funciona de forma parecida com o filter, porém retorna apenas o primeiro valor que satisfizer o teste, podendo ser uma string ou um número.

findIndex()

Funciona igual o `find()`, mas retorna o índice em vez do elemento, possibilitando usá-lo em outras partes do código.

indexOf()

Localiza e retorna o índice referente à primeira ocorrência de determinado valor em um array. Caso não exista nenhuma ocorrência do valor, retorna `-1`.

lastIndexOf()

Funciona da mesma forma que o `indexOf()`, porém retorna o índice referente à *última* ocorrência de um valor em um array, varrendo o array de trás para frente.

forEach()

Executa uma função em cada elemento do array de forma individual.

Não altera o array original e nem retorna um valor, deixando esse trabalho a cargo da função escolhida.

pop()

Retira o último elemento do array.

Altera o array original removendo o elemento.

shift()

Retira o primeiro elemento do array.

Altera o array original removendo o elemento e trocando o índice de todos os elementos para um a menos do que eram, o índice 1 passa a ser o 0, o 2 passa a ser o 1, e assim por diante.

push()

Adiciona o elemento passado como parâmetro do array, porém na última posição.

Altera o array original com o novo valor.

unshift()

Funciona igual ao `push()`, porém adiciona na primeira posição e acaba trocando o índice de todos os elementos.

Altera o array original com o novo valor.

reduce()

Utiliza uma função definida pelo usuário em cada um dos elementos, guardando o resultado em uma variável que pode ser acessada dentro da função que foi definida, retornando um único valor no final, reduzindo o array para um único valor.

reduceRight()

Funciona igual o `reduce()` porém começa do final do array e segue até o início.

reverse()

Inverte a ordem dos elementos do array, então o primeiro vira o último, o segundo o penúltimo e assim por diante.

slice()

Copia uma parte do array para outro array

.

sort()

Organiza o array de acordo com a classificação Unicode, onde os números vêm antes das letras, porém não funciona corretamente para números, onde temos que definir uma função que irá auxiliar o comando.

splice()

Consegue remover, um ou mais elementos consecutivos caso o segundo parâmetro tenha um valor maior que 0, e incluir um ou mais elementos a partir de um índice escolhido.

Manipulando Arrays Vazios (empty)

Esse comportamento de array é chamado de Array Esparso.

Dividindo com Slice( )

O método Slice ( ) pede alguns parâmetros para poder começar a funcionar

1. Número START (onde começará o corte)
2. Número END (onde terminará o corte)

```
const nomes = ['João', 'Juliana', 'Ana', 'Caio', 'Lara', 'Marjorie', 'Guilherme', 'Aline',
'Fabiana', 'André', 'Carlos', 'Paulo', 'Bia', 'Vivian', 'Isabela', 'Vini', 'Renan', 'Renata', 'Daysi',
'Camilo']

// desafio: dividir todos esses nomes em DUAS salas;

const salaOne = nomes.slice(0, nomes.length/2) // vai do índice '0' até a metade:
length/2 (tamanho do array divido por 2)
const salaTwo = nomes.slice(nomes.length / 2) // vai da metade até o FINAL do
array (onde termina)

console.log(`Alunos da SALA 1: ${salaOne}`);
console.log(`Alunos da SALA 2: ${salaTwo}`);
```

Métodos são FUNÇÕES.

Alterando com Splice ( )

Com o método Splice ( ) eu consigo colocar um elemento NO LUGAR do outro. Consigo manipular o array e colocar o elemento novo em QUALQUER lugar que eu quiser.

O splice ele aceita PARAMETROS (índice onde começo a alterar, e O QUE eu vou querer incluir)

```
const listaChamada = ['João', 'Ana', 'Caio', 'Lara', 'Marjorie', 'Leo']
// Ana e Caio => sairam
// Rodrigo => Entrou
// Inserir Rodrigo no seu lugar correto de chamada.
listaChamada.splice(1, 2, 'Rodrigo')
// No lugar 1 e 2 eu adicionei o 'Rodrigo'
console.log(`Lista de chamada: ${listaChamada}`)
```

Porém, o SPLICE pode funcionar sem o SEGUNDO parâmetro, que é um parâmetro que sinaliza o que é para ser adicionado;

concatenando (+) arrays

- ⇨ Tenho 2 Arrays (Array1 e Array2) e quero concatena-los;
- ⇨ Utilizando o método '.concat(..)' no Array1 e passando o Array2 como parâmetro do método concat(Array2)

```
const salaPython = ['Melissa', 'Helena', 'Rodrigo'] //Array1
const salaJavaScript = ['Ju', 'Léo', 'Raquel'] //Array2
const salasUnificadas = salaPython.concat(salaJavaScript)
console.log(salasUnificadas)
```

O método .concat( ) não mexe nas Arrays originais (Array1) e (Array2), por isso que criei uma nova variável no código acima (salasUnificadas) para armazenar meu novo Array Unificado que se formou.

Matrizes com Arrays

```
idades = [30, 35, 28]
nomes = ["Ana", "Juliana", "Leonardo"]
faculdade = [false, true, true]

funcionarios = [nomes,idades,faculdade]
```

O array 'funcionarios' se tornou um array de duas dimensões, por ter outros arrays dentro dele;

concat() é um método útil quando não se deseja alterar o array original, e sim fazer uma cópia alterada. Caso isso não seja necessário, considere utilizar push() ou splice() para inserir novos elementos ou fazer alterações no array original.