

# ARRAYS E FUNÇÕES NATIVAS



DEVinHouse

Parcerias para desenvolver a sua carreira

**SENAI**

<LAB365>

# AGENDA

- Revisão Kahoot -
- Foreach
- Funções nativas de array

# FOREACH

- O **forEach** é uma simplificação do operador **for** para iteração em **arrays**.
- Permite acessar cada elemento individualmente iterando sobre toda a coleção sem a necessidade de informação de índices.

```
array.forEach((element, index, arr) => {  
  });
```

```
var array = [1, 2, 3];  
array.forEach((item) => {  
  console.log('Contador:', item);  
});
```

```
[Running] no  
Contador: 1  
Contador: 2  
Contador: 3
```

# EXERCÍCIO

Em grupos e utilizando o laço de interação foreach execute uma função que irá percorrer a seguinte lista de nomes:

```
var names = ["Ben", "Carol", "Iane", "Jafar", "Matt", "Priya", "Brian", "Guilherme",  
"Bruna"];
```

A cada item deverá ser verificado se o primeiro caractere é B, caso seja, adicione toda a string do item em um outro array.

# INTERVALO DE AULA

## DEV!

Finalizamos o nosso primeiro período de hoje. Que tal descansar um pouco?!

Nos vemos em 20 minutos.

**Início:** 21:00

**Retorno:** 21:20



# MANIPULAÇÃO DE ARRAYS

- **Filter:** Serve para filtrar itens de um array. O `Array.filter` executa a **condição** para **cada item** do array. Se a função retornar um valor `true`, o item permanece na lista, caso retorne `false`, o item é removido da lista.

```
const numeros = [1, 2, 3, 4];  
const pares = numeros.filter((numero) => numero % 2 === 0);  
console.log(pares);
```

[ 2, 4 ]

# EXERCÍCIO

Observe o array ao lado:

Apenas retorne os objetos que possuem a idade maior ou igual a 18 anos usando o `.filter`

```
let array = [  
  {  
    nome: "Rayane",  
    sobrenome: "Cristina",  
    idade: 21  
  },  
  {  
    nome: "Carlos",  
    sobrenome: "Henrique",  
    idade: 17  
  },  
  {  
    nome: "Julio",  
    sobrenome: "Cesar",  
    idade: 19  
  },  
  {  
    nome: "Camila",  
    sobrenome: "Fernandes",  
    idade: 18  
  },  
  {  
    nome: "Julia",  
    sobrenome: "Fernandes",  
    idade: 10  
  },  
  {  
    nome: "Bruno",  
    sobrenome: "Albuquerque",  
    idade: 31  
  },  
  {  
    nome: "Túlio",  
    sobrenome: "Bastos",  
    idade: 22  
  },  
  {  
    nome: "Cristiane",  
    sobrenome: "Maria",  
    idade: 41  
  },  
]
```

# MANIPULAÇÃO DE ARRAYS

- **Find:** Retorna o valor do **primeiro elemento** do array que **satisfizer** a **condição** provida. Caso não encontre nenhum item é retornado **undefined**.

```
const numeros = [1, 2, 3, 4];  
const par = numeros.find((numero) => numero % 2 === 0);  
console.log(par);
```

2



# EXERCÍCIO

Usando o array retornado após o `.filter` no slide 06 imprima em uma tag `h1` qual o nome e sobrenome da primeira pessoa que possua idade maior que 30 anos.

# MANIPULAÇÃO DE ARRAYS

- **Map:** Permite aplicar uma **transformação** para cada elemento do array, **gerando** um **novo array** como resultado.

```
const numeros = [1, 2, 3, 4];  
const numerosVezes2 = numeros.map((numero) => numero * 2);  
console.log(numerosVezes2);
```

[ 2, 4, 6, 8 ]

## EXERCÍCIO - 20 min

Usando o array no slide 06 faça um filtro com o `.filter` mas dessa vez retornando os objetos que possuam idade menor que 18 anos.

Com esse array de objetos filtrado, agora use o `.map` para criar uma nova chave chamada `anosParaMaioridade` sendo seu valor a quantidade de anos que falta para o usuário completar 18 anos.

Dica: `anosParaMaioridade` irá receber `18 - idade`;

# MANIPULAÇÃO DE ARRAYS

- **Some: Testa se ao menos um** dos elementos no array **passa no teste** implementado pela função atribuída e retorna um valor true ou false.

```
const numeros = [1, 2, 3, 4];  
const peloMenos1 = numeros.some((numero) => numero % 2 ===  
0);  
console.log(peloMenos1);
```

true

# MANIPULAÇÃO DE ARRAYS

- **Every:** Testa se **todos** os elementos do array **passam pelo teste** implementado pela função fornecida.

```
const numeros = [1, 2, 3, 4];  
const todos = numeros.every((numero) => numero % 2 == 0);  
console.log(todos);
```

false

# MANIPULAÇÃO DE ARRAYS

- **Sort: Ordena** os **elementos** do próprio array e **retorna** o **array**. A ordenação padrão é de acordo com a pontuação de código unicode.

```
const numeros = [4, 1, 2, 3];  
const ordenado =  
numeros.sort();  
console.log(ordenado);
```

[ 1, 2, 3, 4 ]

```
const numeros = [4, 1, 2, 3];  
const ordenado = numeros.sort((a, b) =>  
a - b);  
console.log(ordenado);
```

[ 1, 2, 3, 4 ]

# EXERCÍCIO

Usando o array de números abaixo ordene em ordem decrescente os seus itens.

Dica: Para inverter a ordem de um array, use o `.reverse()`

```
let array = [4,2,6,5,3];
```

Eis que te perguntam quando seus problemas começaram:



# MANIPULAÇÃO DE ARRAYS

- **Reduce:** Executa uma função para cada item do array, porém o seu **retorno** será a **redução** dos itens internos do array.
- O reduce recebe **dois parâmetros no seu callback**, o **valor anterior** e o **valor atual**. Ainda é possível adicionar como parâmetro do reduce o **valor inicial do retorno do callback**.

```
const array = [1, 2, 3, 4];
const valorInicial = 0;
const soma = array.reduce(
  (valorAnterior, valorAtual) => valorAnterior + valorAtual,
  valorInicial
);
// 0 + 1 + 2 + 3 + 4
console.log(soma); //10
```



# EXERCÍCIO

Observando o array abaixo e usando o reduce, encontre qual é o item que possui o maior valor.

Dica: se o item atual for maior que o valor anterior, o valor anterior irá passar a ser o item atual.

```
const array = [1, 22, 31, 40, 3, 5];
```



map, filter, and reduce  
explained with emoji 🤔

```
map([🐮, 🍌, 🐔, 🌽], cook)  
=> [🍔, 🍟, 🍗, 🍿]
```

```
filter([🍔, 🍟, 🍗, 🍿], isVegetarian)  
=> [🍟, 🍿]
```

```
reduce([🍔, 🍟, 🍗, 🍿], eat)  
=> 🤮
```

# MATERIAL COMPLEMENTAR

- <https://acervolima.com/introducao-ao-es6/>
- [https://docs.google.com/presentation/d/1k1SIMn5c1UuvFnxyTyPGzeEl\\_tAhfMUI1u2rHvoo2ZfM/edit?usp=sharing](https://docs.google.com/presentation/d/1k1SIMn5c1UuvFnxyTyPGzeEl_tAhfMUI1u2rHvoo2ZfM/edit?usp=sharing)
- [https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Functions/Arrow functions](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Functions/Arrow_functions)
- <https://www.rodrigobrito.dev.br/blog/js-0407-javascript-completo-es6-iteracao-em-array>
- [https://www.w3schools.com/js/js\\_versions.asp](https://www.w3schools.com/js/js_versions.asp)

## AVALIAÇÃO DOCENTE

O que você está achando das minhas aulas neste conteúdo?

[Clique aqui](#) ou escaneie o QRCode ao lado para avaliar minha aula.

Sinta-se à vontade para fornecer uma avaliação sempre que achar necessário.





# DEVinHouse

Parcerias para desenvolver a sua carreira

**OBRIGADO!**



<LAB365>