# Tema 09 – Métodos de alto nível para manipular arrays

## Introdução

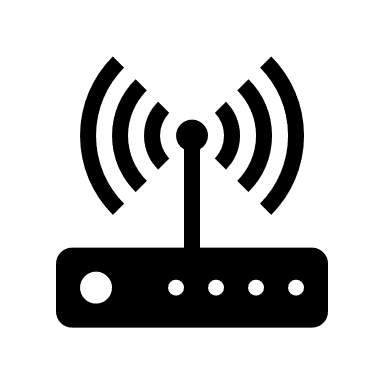
Olá, futuras mentes desenvolvedoras, sejam bem-vindos do curso de Frontend do Instituto da Oportunidade Social. Nessa aula, você aprenderá como utilizar métodos de alto nível forEach(), map(), filter() e find() para manipular arrays.

## Criação do Projeto inicial

Siga os passos para criar o projeto:

1. Abra o VS Code e escolha um diretório de trabalho para o seu projeto.
2. Crie um diretório para seu projeto com o nome representativo, por exemplo, 09\_JS\_Array\_High\_Level.
3. Crie um arquivo dentro do diretório do projeto com o nome index.html.
4. Insira o seguinte código no seu arquivo index.html.

|  |
| --- |
| <!DOCTYPE html>  <html lang="pt-br">      <head>          <meta charset="UTF-8" />  <link rel="shortcut icon" href="#" />          <title>High Level Methods</title>      </head>      <body>          <script src="./js/main.js"></script>      </body>  </html> |

1. Esse código mostra a marcação <script> sem nenhum código JavaScript entre a abertura e o fechamento da tag, apenas o atributo **src** com o valor main.js. Isso significa que o código JavaScript está em um arquivo externo. Portanto, temos que criar esse novo arquivo **main.js** dentro do diretório do projeto.
2. Vamos deixar o arquivo main.js vazio e à medida que vamos aprendendo coisas novas iremos completar a implementação do código JavaScript.
3. Abra o arquivo index.html, clique no botão **Go Live** da extensão **Live Server** e abra as Ferramentas de desenvolvimento do navegador web (Atalho **F12**).

## Método forEach()

O método **forEach()** executa uma determina função para cada um dos elementos de um array. Ele não é executado em elementos vazios do array. A sintaxe do método é:

nome\_do\_array.forEach(callback(currentValue [, index [, array]])[, thisArg]);

Os parâmetros entre colchetes são opcionais e cada um significa:

* callback: é a função que será chamada para ser executada em cada elemento do array.
* currentValue: é o valor do elemento que está sendo processado no momento.
* index (opcional): O índice do elemento atual sendo processado no array.
* array (opcional): O array que forEach() está sendo aplicado.
* thisArg (opcional): Valor a ser usado como this quando executar callback.

Desse modo, o **forEach()** executa uma vez função que estará no lugar o argumento **callback** para cada elemento de array que possui um valor atribuído. Algumas das vantagens do método **forEach()** são:

* O método percorre automaticamente os elementos do array.
* O método permite executar uma função sobre cada elemento do array.

### Vamos praticar

Vamos implementar o código no arquivo **main.js** para ver tudo acontecendo.

1. No arquivo, **main.js** digite o seguinte código.

|  |  |
| --- | --- |
| 1 | // Método forEach()  const frutas = ['apple', 'orange', 'cherry'];  frutas.forEach(minhaFuncao);  function minhaFuncao(item, index) {      console.log(`Índice: ${index}`);      console.log(`Elemento: ${item}`);  } |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |

1. Vamos ver o resultado mostrado no console.

Graphical user interface, text, application, email

Description automatically generated

A instrução na linha 3 mostra como chamar o método **forEach()**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome do array** | **Ponto final** | **forEach** | **(função chamada)** |
| frutas | . | forEach | (minhaFuncao) |

Observe que para cada elemento a função **minhaFuncao** foi executada para cada elemento, imprimindo no console o índice e o valor do elemento.

Importante: Mas quase nunca faremos um código como mostrado anteriormente, isso porque o JS tem **arrow function**, que é muito mais prático e torna o código mais enxuto e elegante. Por isso: o código anterior ficar assim com **arrow function**:

1. No arquivo, **main.js** digite o seguinte código.

|  |  |
| --- | --- |
| 10 | // Método forEach() com Arrow Function  console.clear();  const frutas01 = ['apple', 'orange', 'cherry'];  frutas.forEach((index, item) => {      console.log(`Índice: ${index}`);      console.log(`Elemento: ${item}`);  }); |
| 11 |
| 12 |
| 13 |
| 14 |
| 15 |
| 16 |

1. Vamos ver o resultado mostrado no console.

Graphical user interface, text, application, email

Description automatically generated

O resultado é o mesmo do anterior, mas o código é mais simples. Por isso, entenda bem essa estrutura de chamada de **arrow function**, pois ela é muito útil em JS.

1. Além da praticidade de percorrer array, também podemos aplicar esse método em arrays de objetos.
2. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 18 | // forEach() com array de objetos  console.clear();  const tarefas = [      {          id: 1,          texto: 'Levar o lixo para fora',          isCompleted: true,      },      {          id: 2,          texto: 'Encontrar com o chefe',          isCompleted: true,      },      {          id: 3,          texto: 'Consulta no dentista',          isCompleted: false,      },  ];  tarefas.forEach((teste) => console.log(teste.texto)); |
| 19 |
| 20 |
| 21 |
| 22 |
| 23 |
| 24 |
| 25 |
| 26 |
| 27 |
| 28 |
| 29 |
| 30 |
| 31 |
| 32 |
| 33 |
| 34 |
| 35 |
| 36 |
| 37 |
| 38 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application, email

Description automatically generated

O interessante é que não é necessário indicar o índice do array com nos laços de repetição.

Importante: Não apague o código anterior, pois vamos utilizar o array de objetos **tarefas** nos métodos, que iremos aprender daqui para frente.

## Método map()

O método **map()** cria um novo array com o resultado gerado pela chamada de uma função para cada elemento do array. Ele não é executado em elementos vazios do array. A sintaxe do método é:

let novo\_array = nome\_do\_array.map(callback[, thisArg]);

Os parâmetros entre colchetes são opcionais e cada um significa:

* callback: é a função que o retorno produz o elemento do novo Array.

Desse modo, o **map()** executa uma vez função que estará no lugar o argumento **callback** para cada elemento de array que possui um valor atribuído e constrói um novo array com base nos valores retornados pela execução da função.

### **Vamos praticar**

1. No arquivo, **main.js** digite o seguinte código:

|  |  |
| --- | --- |
| 40 | // Método map()  const numeros = [4, 9, 16, 25];  const newArray = numeros.map(Math.sqrt);  console.clear();  console.log('Números:');  console.log(numeros);  console.log('Novo array:');  console.log(newArray); |
| 41 |
| 42 |
| 43 |
| 44 |
| 45 |
| 46 |
| 47 |

1. O resultado é mostrado no console do navegador.

Graphical user interface, text, application, email

Description automatically generated

O método **map()** o método **sqrt** do objeto **Math**, que retorna a raiz quadrada (**sq**uare **r**oo**t**) de cada elemento do array e armazenado o resultado no novo array com o nome **newArray**, que é impresso no console como array.

1. Vamos ver outro exemplo que utilizará o array de objetos **tarefas** nesse método, que foi usado anteriormente.
2. No arquivo, **main.js** digite o seguinte código.

|  |  |
| --- | --- |
| 49 | // Outro exemplo do map  console.clear();  const mapText = tarefas.map((valor) => {      return valor.texto;  });  //Imprime o novo array  console.log(mapText); |
| 50 |
| 51 |
| 52 |
| 53 |
| 54 |
| 55 |

1. O resultado é mostrado no console do navegador.

Graphical user interface, text, application, email

Description automatically generated

O método **map()** executa a função para armazenar o valor do campo texto de cada objeto do array tarefas. Observe que é criado um novo array com o nome **mapText**, que é impresso no console como array.

## Método filter()

O método **filter()** cria um novo array com o resultado do filtro aplicado pelo método, ou seja, os elementos que “passaram” na condição configurada no filtro. A sintaxe do método é:

let novo\_array = nome\_do\_array.filter(callback[, thisArg]);

Os parâmetros entre colchetes são opcionais e cada um significa:

* callback: é a função com a condição para testar cada elemento do array.

Desse modo, o **filter()** executa uma vez função que estará no lugar o argumento **callback** para cada elemento de array que possui um valor atribuído e constrói um novo array com os elementos que retornarem **true** para a condição configurada no filtro.

### Vamos praticar

1. No arquivo, **main.js** digite o seguinte código.

|  |  |
| --- | --- |
| 57 | // Método filter  console.clear();  const idades = [32, 33, 16, 40];  let filtroIdade = idades.filter((idade) => {      return idade >= 18;  });  console.log(filtroIdade); |
| 58 |
| 59 |
| 60 |
| 61 |
| 62 |
| 63 |
| 64 |

1. O resultado é mostrado no console do navegador.

Graphical user interface, text, application

Description automatically generated

O filtro criado na linha 62 diz para retornar os valores maiores de 18, ou seja, serão retornados os valores 32, 33 e 40. Como podemos ver na mensagem impressa no console.

1. Vamos utilizar o array de objetos **tarefas** nesse método, que foi usado anteriormente.
2. No arquivo, **main.js** digite o seguinte código.

|  |  |
| --- | --- |
| 66 | // Outro exemplo do filter  console.clear();  const filtoCompletas = tarefas.filter((item) => {      return item.isCompleted === true;  });  console.log(filtoCompletas); |
| 67 |
| 68 |
| 69 |
| 70 |
| 71 |
| 72 |

1. O resultado é mostrado no console do navegador.

Graphical user interface, text, application

Description automatically generated

A condição do filtro configurada era para retornar todos os objetos do array que possuíam o campo/chave com o valor **true** (linha 69). Desse modo o resultado é o retorno dos dois primeiros objetos, como pode ser visto no valor impresso. Caso você não visualize os valores do array:



Basta você clicar na seta ao lado do nome Array para expandir os objetos.

## Método find()

O método **find()** retorna o valor dos elementos do array que passarem na condição configurada. Semelhante ao método **filter()**, porém o **find()** não cria um novo array. A sintaxe do método é:

nome\_do\_array.find(callback(element[, index[, array]])[, thisArg]);

Os parâmetros entre colchetes são opcionais e cada um significa:

* callback: é a função com a iteração de cada elemento do array.

Desse modo, o **find()** executa uma vez função que estará no lugar o argumento **callback** para cada elemento de array e retorna os valores que retornarem **true** para a condição configurada no filtro.

### Vamos praticar

1. No arquivo, **main.js** digite o seguinte código.

|  |  |
| --- | --- |
| 74 | // Método find()  console.clear();  const meuArray = [      { nome: 'apples', quantidade: 2 },      { nome: 'bananas', quantidade: 0 },      { nome: 'cerejas', quantidade: 5 },  ];  console.log(      meuArray.find((fruta) => {          return fruta.nome === 'cerejas';      })  ); |
| 75 |
| 76 |
| 77 |
| 78 |
| 79 |
| 80 |
| 81 |
| 82 |
| 83 |
| 84 |
| 85 |
| 86 |

1. O resultado é mostrado no console do navegador.

Graphical user interface, text, application, email

Description automatically generated

Objeto que o método **find()** retornou, o objeto cujo campo nome era igual a cerejas, como foi configurada a condição na linha 84.

## Para Aprender mais

Procure sempre aprender e estudar mais. Seguem alguns links para você estudar e aprender mais:

forEach()

<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach>

<https://www.w3schools.com/jsref/jsref_foreach.asp>

map()

<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/map>

<https://www.w3schools.com/jsref/jsref_map.asp>

filter()

<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/filter>

<https://www.w3schools.com/jsref/jsref_filter.asp>

find()

<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/find>

<https://www.w3schools.com/jsref/jsref_find.asp>

Existem muito outros métodos para manipular arrays, vale a pena você conferir a lista nos links:

[https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global\_Objects/Array#](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array)

<https://www.w3schools.com/jsref/jsref_obj_array.asp>