# Tema 04 – Arrays

## Introdução

Olá, futuras mentes desenvolvedoras, sejam bem-vindos do curso de Frontend do Instituto da Oportunidade Social. Nessa aula, você aprenderá como trabalhar com arrays.

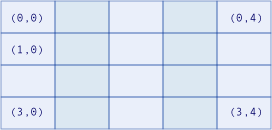
## Arrays

Arrays são usados para armazenar múltiplos valores em uma única variável. Arrays podem ter mais de uma ou mais dimensões e, na literatura em português, é normal você encontrar o nome vetor para referenciar array de uma dimensão e matrizes para array de duas dimensões. Resumindo, você pode chamar tudo de array que qualquer pessoa que programa irá entender.

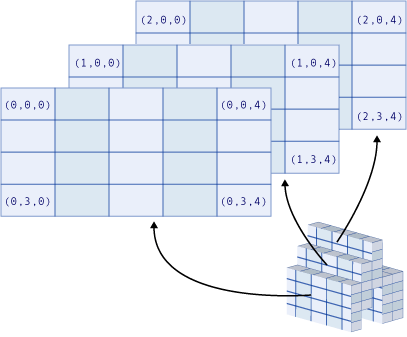
É importante saber que primeiro valor de um array possui índice zero ( 0 ), portanto para array unidimensionais, temos os seguintes índices:

Diagram that shows a one-dimensional array.

Para arrays bidimensionais, temos os seguintes índices:



E para arrays tridimensionais, temos os seguintes índices:



Assim por diante. É muito importante que você saiba isso de cor, por não cometer erros na programação, quando for necessário preencher ou percorrer um array.

Vamos ver isso na prática.

### Vamos praticar

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, 04\_JS\_Arrays.
3. Crie um arquivo dentro do diretório do projeto com o nome index.html.
4. Insira o seguinte código no seu arquivo index.html.

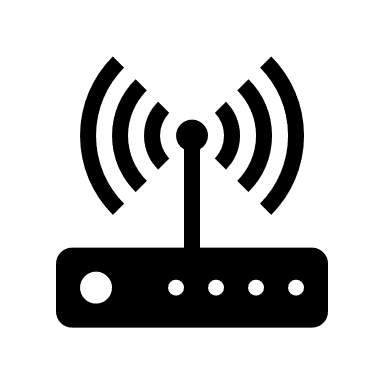
|  |
| --- |
| <!DOCTYPE html>  <html lang="pt-br">      <head>          <meta charset="UTF-8" />  <link rel="shortcut icon" href="#" />          <title>Strings</title>      </head>      <body>  <script src="./js/main.js"></script>      </body>  </html> |

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. Ao criar o seu arquivo a hierarquia do diretório do projeto deverá ficar assim:

Graphical user interface, text, application, chat or text message

Description automatically generated

Observe que mudamos a organização dos arquivos no projeto, agora o arquivo **main.js** está em uma pasta chamada **js**, enquanto o arquivo **index.html** está no diretório raiz do projeto. Isso é importante para você ser uma pessoa organizada com os seus projetos. Além disso, o caminho do script agora está com apontando para o novo diretório (<script src="**./js/main.js**"></script>).

1. Vamos deixar o arquivo main.js vazio e à medida que vamos aprendendo coisas novas vamos completando a implementação do código JavaScript.
2. 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**).

Agora, vamos implementar todo o nosso código desse tema no arquivo main.js e ver as mensagens impressas no console.

1. Vamos ver como acessar elementos armazenados em um array unidimensional. No arquivo, **main.js** digite o seguinte código.

|  |  |
| --- | --- |
| 1 | //Arrays unidimensionais  // Podemos criar array assim  const num1 = new Array(30, -1, 5, 3, 121);  // Ou podemos fazer  const num2 = [-2, 40, 16, 111, 33, 64];  console.log(num1);  console.log(num2);  // Acessando elementos diversos dos arrays  console.log(`primeiro elemento de num1: ${num1[0]}`);  console.log(`primeiro elemento de num2: ${num2[0]}`);  console.log(`terceiro elemento de num1: ${num1[2]}`);  console.log(`quinto elemento de num2: ${num2[5]}`); |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |
| 10 |
| 11 |
| 12 |
| 13 |
| 14 |
| 15 |

1. Antes de explicar o que foi programado vamos ver o resultado mostrado no console.

Graphical user interface, text, application, email

Description automatically generated

As instruções usadas na linha 3 e na linha 5 criam o mesmo tipo de array, a diferença é que a primeira pode exigir mais recurso computacional, portanto é mais lenta, e a segunda é mais prática e mais utilizada na maioria dos casos.

As instruções usadas na linha 3 e na linha 5 criam um array de uma linha e várias colunas, a primeira const num1 = new Array(30, -1, 5, 3, 121)cria um array com 5 colunas, enquanto a segunda const num2 = [-2, 40, 16, 111, 33, 64] cria um array com 6 colunas. Observe que as colunas são definidas pela vírgula que está entre os elementos.

Olha que interessante, como foi falando no início do tema, para acessar o primeiro elemento devemos indicar o índice zero como as instruções nas linhas 11 e 12.

1. Agora vamos ver como trabalhar com arrays bidimensionais. Nesse caso, o primeiro elemento vai estar na posição (0, 0). Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 17 | //Arrays bidimensionais  console.clear();  const matrix = [      ['Banana', 'Maçã', 'Pêra'],      ['Laranja', true, 1],      [null, 'Uva', -350],  ];  // Acessando elementos diversos dos arrays  console.log(`Acessa a primeira linha: ${matrix[0]}`);  console.log(`Acessa o primeiro elemento: ${matrix[0][1]}`); |
| 18 |
| 19 |
| 20 |
| 21 |
| 22 |
| 23 |
| 24 |
| 25 |
| 26 |
| 27 |

Você pode notar que o array em JS não precisa armazenar somente o mesmo tipo, no array matrix, que foi criado, temos tipos literais, numéricos, booleanos ,etc.

Você deve notar também, que um array bidimensional possui diversos array dentro dele, por isso é necessário que os elementos de cada linha devam estar dentro de colchetes [ ].

Outro ponto importante é a instrução matrix[0] acessa toda a linha zero do array, ou seja, quando especificamos uma dimensão em um array bidimensional, estamos buscando a linha inteiro do array. Portanto, para acessar o elemento, devemos indicar as duas dimensões: matrix[0][1].

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application, email

Description automatically generated

1. Para alterar algum valor é bem simples, basta indicarmos o índice do elemento que queremos alterar e igualar ao valor desejado. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 29 | // Alterando um valor do array  console.clear();  let moedas = ['Real', 'Dólar', 'Coroa', 'Peso'];  console.log(`Antes: ${moedas}`);  moedas[2] = 'Yen';  console.log(`Depois: ${moedas}`);  let matrix1 = [      [1, 2, 3],      [4, 5, 6],  ];  console.log('Antes:');  console.log(matrix1);  matrix1[0][2] = -10;  console.log('Depois:');  console.log(matrix1); |
| 30 |
| 31 |
| 32 |
| 33 |
| 34 |
| 35 |
| 36 |
| 37 |
| 38 |
| 39 |
| 40 |
| 41 |
| 42 |
| 43 |
| 44 |
| 45 |
| 46 |

A instrução moedas[2] = 'Yen' muda o valor terceiro elemento do array unidimensional **moeda**. Enquanto, a instrução matrix1[0][2] = -10 altera elemento (0, 2 ) do array bidimensional **matrix1**.

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application

Description automatically generated

## Métodos de arrays

Vamos aprender sobre os métodos de arrays continuando a nossa programação no arquivo main.js.

### Convertendo Array em String

O método **toString()** converte um array em uma string, onde os elementos do array são separados por virgulas. Vamos ver isso na prática, siga os passos para continuar a implementação:

1. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 48 | //Métodos em Arrays  console.clear();  let numArray = [1, 2, 3, 4];  console.log(typeof numArray);  let numArray2 = numArray.toString();  console.log(typeof numArray2); |
| 49 |
| 50 |
| 51 |
| 52 |
| 53 |
| 54 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application, email

Description automatically generated

Observe que o tipo de dados do primeiro array, que foi impresso no console, é um **objeto**, enquanto o tipo de dados do segundo é uma **string**.

Observe também que para acessar um método em JavaScript colocamos o nome\_do\_objeto depois ponto final e depois o nome o método:

|  |  |  |
| --- | --- | --- |
| numArray | . | toString() |
| nome\_do\_objeto | ponto final | nome do método |

Existe também o método **join()** que converte o array em string, mas você pode colocar qualquer caractere como separador dos elementos. Vamos ver na prática:

1. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 57 | // join()  console.clear();  let teste1 = [0, 1, 2, 3, 4, 5];  let teste2 = teste1.join(' \*\*\* ');  console.log(teste2); |
| 58 |
| 59 |
| 60 |
| 61 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, application

Description automatically generated

O resultado impresso no console mostra o separador personalizado inserido entre os elementos do array.

### Propriedade: Tamanho do array

A propriedade **length** retorna tamanho do array, ou seja, a quantidade de elementos que esse array possui. Vamos ver isso na prática, siga os passos para continuar a implementação:

1. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 63 | //Tamanho de Arrays  console.clear();  let array = [0, 1, 2, 3, 4];  console.log(array.length); |
| 64 |
| 65 |
| 66 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application, email

Description automatically generated

Observe que o comando **array.length** retorna a quantidade de elementos do array, que são 5.

### Inserir e remover elemento no array

O método **pop()** remove o último elemento do array e retorna esse elemento retirado, que pode ser armazenado em uma variável. Vamos ver isso na prática, siga os passos para continuar a implementação:

1. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 68 | // Remover e inserir elementos  // pop()  console.clear();  const frutas1 = ['Banana', 'Laranja', 'Maçã', 'Manga'];  let x = frutas1.pop(); // x = "Manga"  console.log(frutas1);  console.log(x); |
| 69 |
| 70 |
| 71 |
| 72 |
| 73 |
| 74 |
| 75 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, application

Description automatically generated

O array criado inicialmente tinha a fruta Manga, após a execução da instrução frutas1.pop(), na linha 72, esse último elemento foi retirado. A variável x foi usada para vermos o elemento retirado sendo armazenado em algum lugar.

Já o método **push()** insere um elemento no final do array e retorna o tamanho do novo array, que pode ser armazenado em uma variável. Vamos ver isso na prática, siga os passos para continuar a implementação:

1. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 77 | // push()  console.clear();  const frutas2 = ['Banana', 'Laranja', 'Maçã', 'Manga'];  let y = frutas2.push('Kiwi'); // y = 5  console.log(frutas2);  console.log(y); |
| 78 |
| 79 |
| 80 |
| 81 |
| 82 |
| 83 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application

Description automatically generated

O array criado inicialmente não continha a fruta Kiwi, após a execução da instrução frutas2.push('Kiwi'), na linha 80, esse elemento foi adicionado na última posição. A variável y foi usada para vermos o tamanho do novo array, agora com o elemento adicionado.

Por sua vez, o método **shift()** remove o elemento no início do array deslocando todos os elementos em uma posição e retorna esse elemento retirado, que pode ser armazenado em uma variável. Vamos ver isso na prática, siga os passos para continuar a implementação:

1. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 85 | // shift()  console.clear();  const frutas3 = ['Banana', 'Laranja'];  let z = frutas3.shift(); // z = 'Banana'  console.log(frutas3);  console.log(z); |
| 86 |
| 87 |
| 88 |
| 89 |
| 90 |
| 91 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application, email

Description automatically generated

O array originalmente tinha dois elementos e depois da execução da instrução frutas3.shift() na linha 88 passou a ter um. A variável z foi usada para vermos o elemento retirado sendo armazenado em algum lugar.

Por fim, o método **unshift()** insere um elemento no início do array e retorna o tamanho do novo array, que pode ser armazenado em uma variável. Vamos ver isso na prática, siga os passos para continuar a implementação:

1. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 93 | // unshift()  console.clear();  const frutas4 = ['Banana', 'Laranja'];  let w = frutas4.unshift('Kiwi'); // w = 3  console.log(frutas4);  console.log(w); |
| 94 |
| 95 |
| 96 |
| 97 |
| 98 |
| 99 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application

Description automatically generated

Observe que os elementos do array forma deslocados para inserir o elemento Kiwi. após a execução da instrução frutas4.unshift('Kiwi'), na linha 96, esse elemento foi adicionado na última posição. A variável w foi usada para vermos o tamanho do novo array, agora com o elemento adicionado.

### Apagando um elemento em uma posição específica

O comando **delete** apaga um elemento na posição desejada e altera para **slot vazio**, que é o mesmo de **undefined**. Vamos ver isso na prática, siga os passos para continuar a implementação:

1. Continuando a implementação do projeto, insira o seguinte código no arquivo main.js:

|  |  |
| --- | --- |
| 101 | // delete  console.clear();  const frutas5 = ['Banana', 'Laranja', 'Maçã', 'Manga'];  delete frutas5[2];  console.log(frutas5); |
| 102 |
| 103 |
| 104 |
| 105 |
| 106 |

1. O resultado é mostrado no console do navegador:

Graphical user interface, text, application

Description automatically generated

Note que o elemento Maçã, que existe quando o array foi criado, foi substituído por <1 empty slot> após a execução da instrução delete frutas5[2].

## Conclusão

Existem muitos outros métodos para manipular arrays e gastaríamos muito tempo se quisessem abordar todos esses métodos. À medida que outros métodos forem aparecendo, vamos explicando cada uma desses. O importante é que você sempre deve usar a internet para procurar soluções em programação, mas não apenas copie e cole uma solução, um bom programador procura entender a solução proposta e adaptá-la no seu projeto.

Para saber mais: Você pode consultar a lista de métodos de strings nos links:

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

<https://www.w3schools.com/js/js_array_methods.asp>

Como desafio, tente entender e implementar alguns desses métodos.