

A horizontal bar with a blue segment on the left and an orange segment on the right.

4954/Programador Web

Qualificação Profissional

Eixo Tecnológico: Informação e Comunicação
Segmento: Tecnologia da Informação (TI)



Prof. Davi Gomes

A horizontal bar with a blue segment on the left and an orange segment on the right.

Introdução: Sintaxe JavaScript, Parte II

O objetivo desta unidade é aprender aspectos adicionais da linguagem JavaScript e começar a escrever programas mais complexos.

Após esta unidade, você será capaz de:

- Ler e escrever sintaxe JavaScript para arrays, loops, objetos e iteradores
- Depurar código JavaScript e analisar mensagens de erro
- Resolva desafios de código relacionados à sintaxe recém-aprendida

Arrays

Arrays

Organizar e armazenar dados é um conceito fundamental da programação. Uma maneira de organizar os dados na vida real é fazendo listas. Vamos fazer um aqui:

Resoluções de Ano Novo:

- Manter um diário

- Fazer um curso de falcoaria

- Aprender a fazer malabarismos






Arrays


Vamos agora escrever esta lista em JavaScript, como um *array*, veja ao lado:

Arrays são a maneira do JavaScript criar listas. Arrays podem armazenar quaisquer tipos de dados (incluindo strings, números e booleanos). Assim como as listas, os arrays são ordenados, o que significa que cada item tem uma posição numerada.

Ao lado, um array com os conceitos que serão abordados:



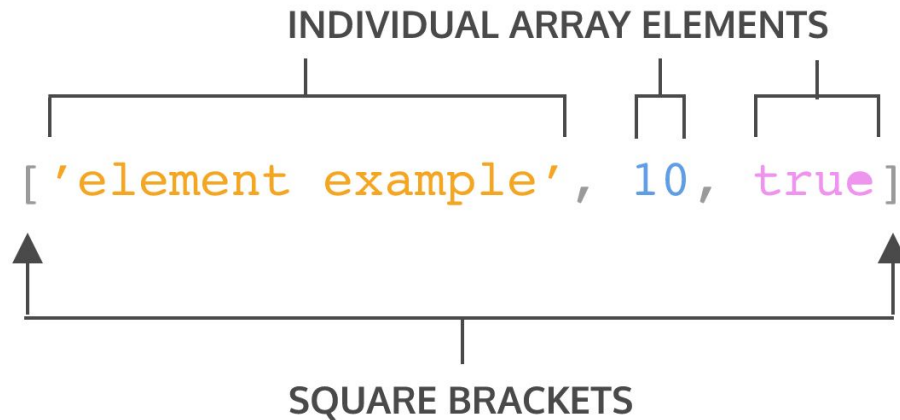
```
let promessasAnoNovo = ['Criar um diário',  
                        'Fazer curso de Falcoaria',  
                        'Aprender a fazer malabarismo'];
```



```
let conceitos = ['criação de arrays',  
                'estrutura de arrays',  
                'manipulação de arrays'];
```

Criando um Array

Uma maneira de criar uma array é usar um *array literal*. Um array literal cria um array ao envolver itens através de colchetes `[]`. Lembre-se do exercício anterior, arrays podem armazenar qualquer tipo de dados — podemos ter um array que contém todos os mesmos tipos de dados ou um array que contém diferentes tipos de dados.

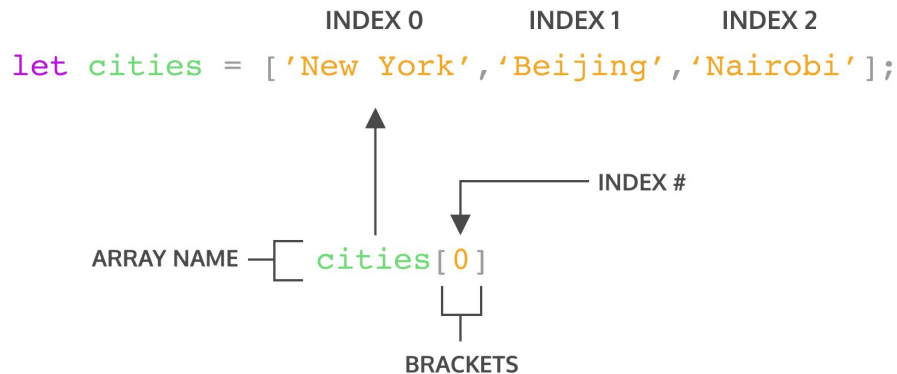


Acessando elementos

Cada elemento em um array possui uma posição numerada chamada de *índice*. Podemos acessar itens individuais usando seu índice, que é semelhante a referenciar um item em uma lista com base na posição do item.

Arrays em JavaScript são indexados por zero, o que significa que as posições começam a contar em 0 e não 1. Portanto, o primeiro item em uma matriz estará na posição 0.

Vejamos como poderíamos acessar um elemento em um array:





Atualizando elementos

Uma vez que aprendemos como acessar elementos em um array através do índice, podemos agora atualizar também o seu valor.

No exemplo ao lado, o array `season` continha os nomes das quatro estações.

No entanto, decidimos que preferíamos dizer *'Autumn'* em vez de *'Fall'*.




```
let seasons = ['Winter', 'Spring', 'Summer', 'Fall'];  
  
seasons[3] = 'Autumn';  
console.log(seasons);  
//Output: ['Winter', 'Spring', 'Summer', 'Autumn']
```




Arrays com let e const

Você deve se lembrar que pode declarar variáveis com as palavras-chave `let` e `const`. As variáveis declaradas com `let` podem ser reatribuídas.



```
let condimentos = ['Ketchup',  
                  'Mostarda',  
                  'Molho de Soja',  
                  'Maionese'];  
  
const utensílios = ['Garfo',  
                   'Faca',  
                   'Pauzinhos',  
                   'Garfo'];
```



Propriedade .length

Uma das propriedades internas de um array é *.length* (tamanho) que retorna o número de itens no array. Acessamos a propriedade *.length* da mesma forma que fazemos com strings.



```
const newYearsResolutions = ['Keep a journal',  
                              'Take a falconry class'];  
  
console.log(newYearsResolutions.length);  
// Output: 2
```



O método .push()

JavaScript integrados que facilitam o trabalho com arrays. Esses métodos são chamados especificamente em arrays para tornar tarefas comuns, como adicionar e remover elementos, mais simples.

Um método, *.push()*, nos permite adicionar itens ao final de um array.



```
const itemTracker = ['item 0', 'item 1', 'item 2'];  
  
itemTracker.push('item 3', 'item 4');  
  
console.log(itemTracker);  
// Output: ['item 0', 'item 1', 'item 2', 'item 3', 'item 4'];
```



O método `.pop()`

Outro método de array, `.pop()`, remove o último item de um array.

- No exemplo ao lado, chamando `.pop()` o array `newItemTracker` removeu o item 2 do fim do array.
- `.pop()` não aceita nenhum argumento, simplesmente remove o último elemento de `newItemTracker`.
- `.pop()` retorna o valor do último elemento. No exemplo, armazenamos o valor retornado em uma variável `removed` para ser utilizada posteriormente.
- `.pop()` é um método que altera o array inicial.



```
const newItemTracker = ['item 0', 'item 1', 'item 2'];

const removed = newItemTracker.pop();

console.log(newItemTracker);
// Output: [ 'item 0', 'item 1' ]
console.log(removed);
// Output: item 2
```



Outros métodos de Array

Existem muito mais métodos de array do que apenas `.push()` e `.pop()`. Você pode ler sobre esses métodos de array acessando o [Docs para JavaScript Arrays](#).

JS

Array Methods

- `toString()`
- `join()`
- `concat()`
- `splice()`
- `slice()`
- `indexOf()`
- `lastIndexOf()`
- `flat()`
- `forEach()`
- `map()`
- `filter()`
- `reduce()`
- `some()`
- `every()`
- `find()`
- `findIndex()`
- `sort()`



Array e Funções

Bem, o que acontece se tentarmos alterar um array dentro de uma função? O array mantém a alteração após a chamada da função ou tem escopo dentro da função?

```
const flores = ['rosa', 'girassol', 'tulipa'];

function addFlor(arr) {
  arr.push('margarida');
}

addFlor(flores);

console.log(flores); // Output: ['rosa', 'girassol', 'tulipa', 'margarida']
```



Array e Funções

Bem, o que acontece se tentarmos alterar um array dentro de uma função? O array mantém a alteração após a chamada da função ou tem escopo dentro da função?

Dê uma olhada no exemplo a seguir, onde chamamos `.push()` de um array dentro de uma função. Lembre-se, o método `.push()` muda, ou altera, um array:

```
const flores = ['rosa', 'girassol', 'tulipa'];

function addFlor(arr) {
  arr.push('margarida');
}

addFlor(flores);

console.log(flores); // Output: ['rosa', 'girassol', 'tulipa', 'margarida']
```



Array e Funções

Vamos repassar o que aconteceu no exemplo:

- O vetor `flores` que tem 3 elementos.
- A função `addFlor()` tem um parâmetro chamado `arr` utiliza o método `.push()` para adicionar um elemento `'margarida'` em `arr`.
- Chamamos `addFlor()` com o argumento `flores` que executará o código que existe dentro de `addFlor()`.

Verificamos o valor de `flores` se agora inclui o elemento `'margarida'`! O array foi modificado!

```
const flores = ['rosa', 'girassol', 'tulipa'];

function addFlor(arr) {
  arr.push('margarida');
}

addFlor(flores);

console.log(flores); // Output: ['rosa', 'girassol', 'tulipa', 'margarida']
```

Revisão

