

FrontEnd

Tipos de variáveis

Em JavaScript, contamos como 3 tipos de variáveis:

Let: São variáveis locais, que estão disponíveis naquele bloco de instrução.

Const: recomendados para valores fixos, que não permitem edições ao longo da execução.

Var: tem a característica de ser global, mas deixou de ser uma boa pratica depois do ES6.

Tipos de variáveis

Em JavaScript, contamos como 3 tipos de variáveis:

Let: São variáveis locais, que estão disponíveis naquele bloco de instrução.

Const: recomendados para valores fixos, que não permitem edições ao longo da execução.

Var: tem a característica de ser global, mas deixou de ser uma boa pratica depois do ES6.

FrontEnd - JavaScript

LET:

Quando pensamos em manipulação de variáveis podemos pensar no **let**.

Porém as variáveis que criarmos dentro de funções, ficarão disponíveis apenas neste contexto.

Para obter o resultado dela, devemos coloca-las no retorno e atribuir esse resultando em outra variável.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    let variavelGlobal = 100;

    function variaveis(){
      let variavelLocal = 20;
      let variavelLocalGlobal = variavelGlobal + 10;
      return {variavelLocal, variavelLocalGlobal};
    }

    let resultado = variaveis();
    alert("Variável Global: " + variavelGlobal);
    alert("Variável Local: " + resultado.variavelLocal);
    alert("Variável Local Global: " + resultado.variavelLocalGlobal);

  </script>
</body>
</html>
```

FrontEnd - JavaScript

CONST:

Usada para valores (estrutura) fixas.

Essa variável se mantém imutável durante toda a execução do sistema, e caso haja tentativa de fazer qualquer movimentação, é dado um erro.

```
alert('variavel Local Global. +  
  
const desconto = 0.01  
  
let preco = 99.59  
desconto = preco*desconto  
alert(desconto)
```

```
Uncaught TypeError: Assignment to constant variable.  
at Ayla05-DemosReact.html:27:18
```

FrontEnd - JavaScript

Coleções:

São um conjunto de valores armazenados em uma única variável.

A manipulação desse array, também tem suas próprias características e ferramentas.

Atavés de chaves, e separados pela virgula, consigo delimitar os valores inseridos.

```
let numeros = [1, 2, 3, 4, 5];  
let nomes = ['Alice', 'Bob', 'Charlie'];
```

FrontEnd - JavaScript

Coleções - Visualização:

Para exibir o índice de todos os itens de uma coleção, por exemplo:

```
let numeros = [10, 20, 30, 40, 50];  
for (let indice in numeros){  
  let a = indice  
  alert(a)  
}
```

Coleções - Visualização:

Para exibir o valor de todos os itens de uma coleção, por exemplo:

```
for (let valor of numeros){  
  let a = valor  
  alert(a)  
}
```

```
numeros.forEach(function(valor){  
  alert(valor)  
})
```

```
nomes.forEach(item=>alert(item))
```

FrontEnd - JavaScript

Inserindo um valor ao **fim** da lista

```
numeros.push(60)  
alert(numeros)
```

Inserindo um valor no **início** da lista

```
numeros.unshift(1)  
alert(numeros)
```


FrontEnd - JavaScript

Inserindo um valor ao **fim** da lista

```
numeros.push(60)  
alert(numeros)
```

Inserindo um valor no **início** da lista

```
numeros.unshift(1)  
alert(numeros)
```

FrontEnd - JavaScript

Excluindo o último valor

```
//alert(numeros)  
numeros.pop()  
alert(numeros)
```

Excluindo o primeiro elemento do array

```
numeros.shift()  
alert(numeros)
```

FrontEnd - JavaScript

Removendo através de índice

```
console.log(numeros)
const indice = 3;
const quantidade = 4;
numeros.splice(indice, quantidade);
console.log(numeros)
```

A partir do 3 item do array, excluir 4 itens do conjunto.

Lembrando que o índice se inicia em 0. A 3ª posição é o de valor 40.

Então os itens 40,50,60,70 são excluídos.

```
► (10) [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
► (6) [10, 20, 30, 80, 90, 100]
```

FrontEnd - JavaScript

Removendo através de índice

```
console.log(numeros)
const indice = 3;
const quantidade = 4;
numeros.splice(indice, quantidade);
console.log(numeros)
```

A partir do 3 item do array, excluir 4 itens do conjunto.

Lembrando que o índice se inicia em 0. A 3ª posição é o de valor 40.

Então os itens 40,50,60,70 são excluídos.

```
► (10) [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
► (6) [10, 20, 30, 80, 90, 100]
```

FrontEnd - JavaScript

Map()

Para que eu consiga manipular os dados de um array, eu posso usar o comando “Map()”, que percorre todo o array, e cria uma variável nova como resultado da expressão “callback”.

```
let numerosDobrados = numeros.map(function(numero) {  
  return numero * 2;  
});
```

```
let resultado = numeros.map(n => n * 2)
```

```
let nomes = ['joão', 'maria', 'pedro', 'ana'];  
  
// Usando o método map() para criar um novo array com nomes em maiúsculas  
let nomesMaiusculos = nomes.map(function(nome) {  
  return nome.toUpperCase();  
});  
  
// Exibindo o novo array  
console.log(nomesMaiusculos); // Saída: ['JOÃO', 'MARIA', 'PEDRO', 'ANA']
```

FrontEnd - JavaScript

Filter

De acordo com os dados de um array, eu posso usar o comando “Filter()”, que percorre todo o array, e cria uma variável nova como resultado da com o resultado do filtro:

```
// Usando o metodo filter() para criar um novo array co  
let numerosPares = numeros.filter(function(numero) {  
    return numero % 2 === 0;  
});
```

```
});  
let numerosMaiores = numeros.filter(function(numero) {  
    return numero > 50;  
});  
alert(numerosMaiores)
```

FrontEnd - JavaScript

Filter

De acordo com os dados de um array, eu posso usar o comando “Filter()”, que percorre todo o array, e cria uma variável nova como resultado da com o resultado do filtro:

```
// Usando o metodo filter() para criar um novo array co  
let numerosPares = numeros.filter(function(numero) {  
    return numero % 2 === 0;  
});
```

```
});  
let numerosMaiores = numeros.filter(function(numero) {  
    return numero > 50;  
});  
alert(numerosMaiores)
```

FrontEnd - JavaScript

Find

Percorre o array até localizar o primeiro item que satisfaça uma dada condição

```
let primeiroDivisivelPorTres = numeros.find(function(numero) {  
    return numero % 3 === 0;  
});  
  
alert(primeiroDivisivelPorTres); // Saída: 12
```


FrontEnd - JavaScript

Find

Percorre o array até localizar o primeiro item que satisfaça uma dada condição

```
let primeiroDivisivelPorTres = numeros.find(function(numero) {  
    return numero % 3 === 0;  
});  
  
alert(primeiroDivisivelPorTres); // Saída: 12
```

FrontEnd - JavaScript

Find

Percorre o array até localizar o primeiro item que satisfaça uma dada condição

```
let primeiroDivisivelPorTres = numeros.find(function(numero) {  
    return numero % 3 === 0;  
});  
  
alert(primeiroDivisivelPorTres); // Saída: 12
```

FrontEnd - JavaScript

Desestruturação

Permite que façamos uma extração dos dados dos arrays ou objetos com o uso de variáveis distintas.

```
//caminho natural
const livros = ['HTML', 'CSS', 'React']
alert(livros[2])

//desestruturação
let[css, html, react]=livros
alert(css)
```

FrontEnd - JavaScript

Desestruturação

Permite que façamos uma extração dos dados dos arrays ou objetos com o uso de variáveis distintas.

```
//caminho natural
const livros = ['HTML', 'CSS', 'React']
alert(livros[2])

//desestruturação
let [css, html, react]=livros
alert(css)
```

FrontEnd - JavaScript

Spread

O operador spread permite que trafeguemos o array como um todo para poder manipular da forma mais adequada, por exemplo, se quisermos agrupar 2 arrays em um só uma boa prática seria:

```
livros1 = ['Harry Potter e o calice de fogo', 'Harry Potter e a Ordem da Fenix']  
livros2 = ['Harry Potter e o enigma do principe', 'Harry Potter e as reliquias da morte']  
livros = [...livros1, ...livros2]  
alert(livros)
```

Se eu quisesse copiar uma array para outra seria:

```
const array1 = [1, 2, 3];  
const array2 = [...array1]; // Copia os elementos de array1 para array2
```

FrontEnd - JavaScript

Spread

Também consigo espalhar esses valores, considerando os acréscimos da instrução, por exemplo:

```
const numeros2 = [...numeros, 60, 70, 80, 90]  
console.log(numeros2)
```

```
// Sem o spread considera o array numeros e não seus valores separadamente  
const numeros3 = [numeros, 60, 70, 80, 90]  
console.log(numeros3)
```

```
▼ (9) [10, 20, 30, 40, 50, 60, 70, 80, 90] ⓘ  
  0: 10  
  1: 20  
  2: 30  
  3: 40  
  4: 50  
  5: 60  
  6: 70  
  7: 80  
  8: 90  
  length: 9  
  ► [[Prototype]]: Array(0)  
  
▼ (5) [Array(5), 60, 70, 80, 90] ⓘ  
  ► 0: (5) [10, 20, 30, 40, 50]  
    1: 60  
    2: 70  
    3: 80  
    4: 90  
    length: 5
```

FrontEnd - JavaScript

1. Desenvolva um programa que a partir de um vetor de inteiros com 5 valores inicializados na declaração apresente o dobro de cada valor armazenado.
2. Desenvolva um programa que a partir de um vetor de inteiros com 8 valores inicializados na declaração apresente a média aritméticas desses valores.
3. Desenvolva um programa que leia a idade de 20 pessoas e apresente as idades acima da média.
4. Desenvolva um programa que leia 10 números e apresente os valores pares.
Caso não tenha nenhum número par apresente a mensagem “Todos os números são ímpares.”
5. Desenvolva um programa que leia 8 números garantindo que os valores informados estejam entre 100 e 200(caso não esteja presente uma mensagem de “valor inválido”). Depois de preenchido apresente os valores armazenados.

FrontEnd - JavaScript

6. Desenvolva um programa que a partir de um vetor de Strings com 10 nomes inicializados na declaração leia um novo nome e verifique se ele está armazenado no vetor, se estiver, apresente a posição (índice) onde ele está, caso contrário, apresente a mensagem “Nome não encontrado!”
7. Desenvolva uma nova versão do programa anterior em que o usuário terá a quantidade de tentativas limitada a 5. Caso o nome seja encontrado apresente a posição (índice) onde ele está e em qual tentativa ele foi encontrado.
8. Desenvolva um programa que leia 6 números inteiros e armazene em um vetor A. Carregue um vetor B (de mesmo tipo e tamanho) com a metade dos valores armazenados em A. Apresente os valores dos dois vetores.
9. Desenvolva um programa que leia 5 números inteiros e armazene em um vetor A. Leia 5 números inteiros e armazene em um vetor B. Carregue e apresente um vetor C com os valores de A e B alternados.