

# ESTRUTURAS DE REPETIÇÃO



DEVinHouse

Parcerias para desenvolver a sua carreira

**SENAI**

<LAB365>

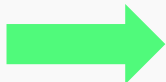
# AGENDA

- Array
- Estruturas de repetição
- laço For
- For of
- While
- Do While

# ARRAYS

- Sempre que precisamos criar uma lista de elementos, utilizamos o tipo de dado Array.
- Os colchetes indicam que a estrutura é um array.
- O Array no JavaScript é heterogêneo (aceita mais de um tipo de dado em uma mesma estrutura).

```
var lista = [  
  "Playstation 5",  
  12,  
  false  
]
```

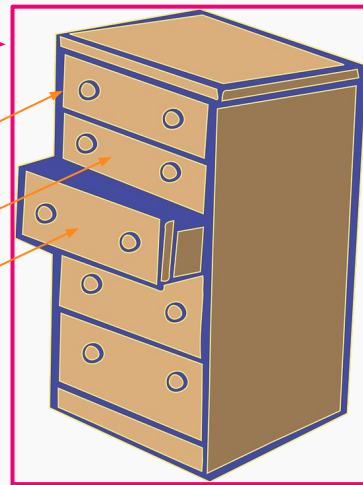


variável do tipo  
array (lista) →

item 0 (Playstation 5)

item 1 (12)

item 2 (false)



Obs: o primeiro índice (posição) do array sempre é 0.

# ARRAYS

- Para acessar cada item de um array indique qual a variável que recebe o array seguido de "[" o índice que deseja acessar e "]".

Exemplo:

```
var listaDesejos = [ "Macbook Pro M1 Max", "Bicicleta Caloi", 12, true ]
```

```
listaDesejos[ 0 ] // "Macbook Pro M1 Max"
```

```
listaDesejos[ 1 ] // "Bicicleta Caloi"
```

```
listaDesejos[ 3 ] // true
```

```
listaDesejos[ 4 ] // undefined
```

Para adicionar ou remover elementos do nosso array e verificarmos a quantidade de elementos, utilizamos **métodos e propriedades de array**:

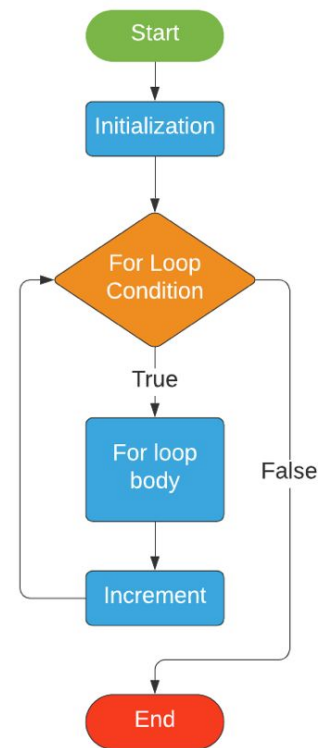
- **array.length**: retorna um number com o tamanho do array (quantidade de elementos)
- **array.push ( *elemento* )**: adiciona *elemento* ao final do array
- **array.pop ( )**: remove o último elemento do array e retorna seu valor

Alguns outros métodos úteis:

- **array.shift( )**: remove o **primeiro** elemento do array e retorna seu valor
- **array.unshift ( elemento )**: adiciona *elemento* no **início** do array e retorna o **length** atualizado
- **array.slice ( inicial, final )**: retorna um “pedaço” do array, a partir do índice *inicial* até (mas não incluindo) o índice *final*.
- **array.join ( separador )**: junta todos os elementos de um array em uma única string, separadas pelo *separador* (por padrão é uma vírgula).

# ESTRUTURA DE REPETIÇÃO

- Quando temos uma determinada cadeia de informações e quisermos **percorrer** essa cadeia executando determinadas ações para cada item dela, é utilizada uma estrutura de repetição.
- É melhor recomendado um laço de repetição para quando quiser executar **n** vezes um bloco de código.
- Exemplos: **for, foreach, while, do while...**



# INTERVALO DE AULA

## **DEV!**

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

Nos vemos em 20 minutos.

**Início:** 20:50

**Retorno:** 21:10






# FOR

## Fluxo:

- Uma **variável** é inicializada com um **valor inicial**.
- Essa variável é utilizada para controlar a **quantidade de vezes** em que o bloco do for **será executado**.
- Ao final do conjunto de comandos a **variável** sempre sofrerá uma alteração, **aumentando** ou **diminuindo** de acordo com a lógica utilizada.

```
for (i = 0; i < 5; i++) {  
    console.log("contador:" + i);  
}
```



```
[Running] · node  
contador:0  
contador:1  
contador:2  
contador:3  
contador:4
```

variável inicializada

condição de parada

Incremento que irá decidir como a condição será finalizada.

Blocagem que será repetida n vezes

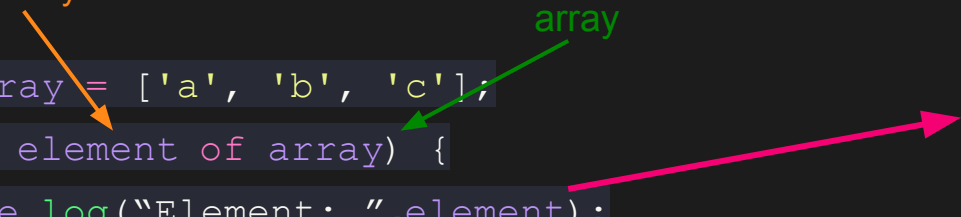
```
for (i = 0; i < 5; i++) {  
    console.log("contador:" + i);  
}
```

# FOR OF (Farofa)

- Este tipo de laço é recomendado quando há interação de propriedades de um array na ordem de inserção original.
- O bloco de código do array será executado para cada item daquele array.

Variável local criada para receber a cada rodada um item do array.

```
const array = ['a', 'b', 'c'];  
for (var element of array) {  
    console.log("Element: ", element);  
}
```



Saída no console

Elemento:	a
Elemento:	b
Elemento:	c

# EXERCÍCIO

Crie um código usando o laço for para percorrer de 1 até 10 mas que apenas os números pares sejam impressos no document.write

Exemplo de saída:

<i>r</i>
2
4
6
8
10

# WHILE

- O **While** (enquanto) executa suas instruções, desde que uma **condição** especificada seja avaliada como verdadeira;
- O **teste** da condição ocorre **antes** que o **laço seja executado**. Desta forma se a condição for verdadeira o laço executará e testará a condição novamente. Se a condição for falsa o laço termina.

```
while ([condicao]) {  
    //bloco  
}
```

```
var contador = 0;  
while (contador < 2) {  
    console.log('contador:', contador);  
    contador++;  
}
```

[Running] - no  
contador: 0  
contador: 1

# EXERCÍCIO

- Visto que para adicionar um item a um array é utilizado o método push(), iremos utilizar para executar o seguinte código.
- O intuito é montar um array apenas com valores ímpares.
- Antes crie um bloco while que irá repetir 10 vezes, a cada rodada ele deve verificar se a rodada atual é ímpar, se for, adicionar no array o valor da rodada atual.
- Ao terminar o bloco while imprima no console o array.

Observe a saída:

```
► (5) [1, 3, 5, 7, 9]
```

# DO...WHILE

- O **Do... While** (faça enquanto) executa suas instruções, desde que uma **condição** especificada seja avaliada como verdadeira;
- O **teste** da condição **ocorre depois** que o **laço seja executado**. Desta forma após a execução se a condição for verdadeira o laço executará novamente. Se a condição for falsa o laço termina.

```
do {  
  
} while ([condicao]);
```

```
var contador = 2;  
do {  
    console.log("contador:", contador);  
    contador++;  
} while (contador < 2);
```

# MATERIAL COMPLEMENTAR

- [https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Building\\_blocks/conditionals](https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Building_blocks/conditionals)
- <https://www.todoespacoonline.com/w/2014/04/estruturas-condicionais-javascript/>
- [https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Loops\\_and\\_iteration](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Loops_and_iteration)
- <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/for...in>
- [Diferença entre for, while e do... while](#)



## 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>