

JavaScript básico





<title>

AGENDA DO </> </> </> MÓDULO

</title>



Agenda do módulo



Anotações



{Booleanos}

{Estrutura IF/ELSE}

{Switch case}



Um exemplo clássico de algoritmo é a **troca de uma lâmpada**. Podemos definir esse algoritmo em poucos passos a serem executados um após o outro.

- O1** Posicionar a escada
- O2** Pegar a nova lâmpada
- O3** Subir na escada
- O4** Remover a lâmpada antiga
- O5** Inserir a lâmpada nova
- O6** Descer da escada
- O7** Descartar a lâmpada antiga

Você trocaria uma lâmpada que estivesse funcionando perfeitamente?

Você pode pegar qualquer lâmpada?
Podemos usar uma lâmpada led ou uma fluorescente?

No mundo real, há diversas condições para que essa lâmpada seja trocada, não é mesmo?

Para que nosso código seja realmente útil, precisamos planejá-lo para que considere essas condições. Isso é feito por meio de estruturas de controle condicionais e de repetição.

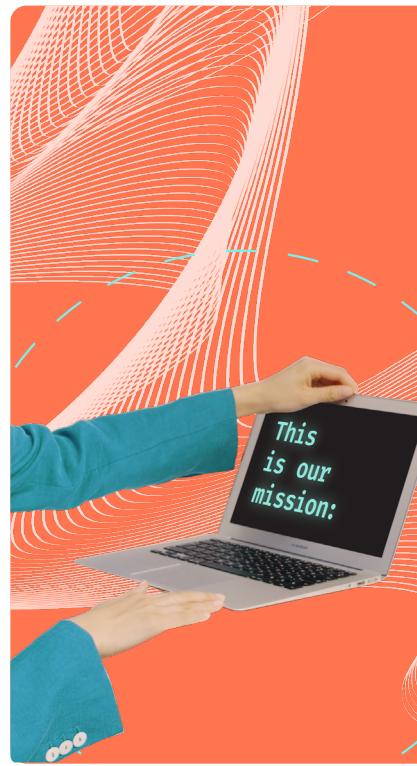


<title>

MISSÃO DO MÓDULO

</title>

Aplicar as **estruturas de controle** na programação em JavaScript.



JavaScript Básico: Estruturas de controle



{01.}

Booleanos

Os valores booleanos são usados em sua maioria em estruturas de controle no JavaScript.

Lógica: Booleanos

Verdadeiro (true)

Falso (false)

Além do verdadeiro e falso, há dois conceitos complementares a eles no JavaScript: **Falsy** e **Truthy**.

Valores falsy são aqueles que não necessariamente são falsos, mas se tornarão quando tentarmos convertê-los, traduzi-los, em um booleano.

Em JavaScript podemos encontrar 5 valores que são “falsy”. Eles são:

- > 0 (zero)
- > “ (string vazia)
- > undefined
- > null
- > NaN

Se esses cinco valores citados têm o valor de falso quando convertidos para um valor booleano, seguindo por essa lógica, todos os outros valores são “Truthy”.

Isso significa que, quando convertidos para booleano, trazem consigo um valor de verdadeiro, de true, como:

- > O strings
- > números
- > arrays (mesmo que vazios)
- > objetos (mesmo que vazios) e funções



**Se as condições
estiverem certas,
programe. Senão,
transforme-as!**

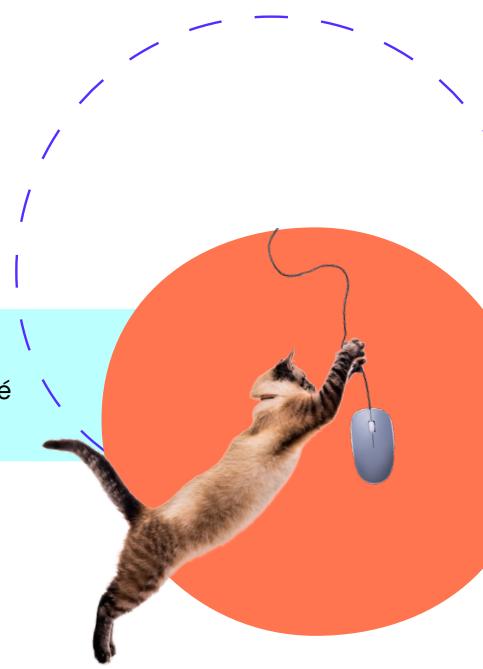


<title>

PULO DO GATO

</title>

- Apesar de poder utilizar as duas formas de comparação (`==` e `===`), é recomendável o uso de 3 sinais.

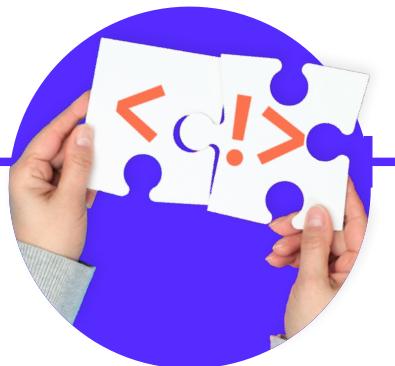




<title>

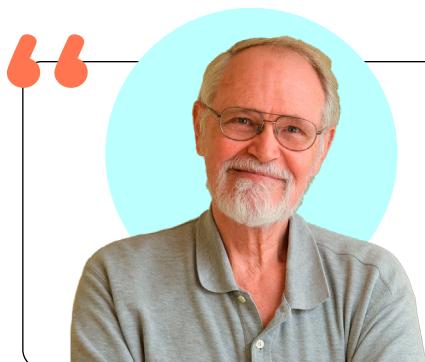
MÃO + + NA MASSA

</title>



Crie uma variável com a sua idade.

- › Em seguida, compare esta variável com outra variável e o tipo que você quiser, mas que no final retorne estes resultados: true, false, false, true.
- › Na primeira rodada de comparações, utilize “==”, e na segunda, “===”.



+

+

+

+

+

+

Controlar a complexidade é a essência da programação de computadores.

Brian Kernighan

{02.}

Estrutura IF/ELSE

Traduzindo ao pé da letra, significa exatamente “Se” e “Senão”, ou seja, estamos dizendo para o código: “Se esse pedaço de código for verdadeiro, faça isso, senão, faça aquilo”. Estamos criando diferentes fluxos de execução.

A sintaxe do If é:

```
if (condição) {  
    //bloco de código  
}
```

Se essa condição que definimos entre parênteses for satisfatória, o bloco de código ali de dentro será executado. Senão, ele será ignorado e código continuarará para o que vem depois.



Quando incluímos o else, teremos essa outra

```
if (condição) {  
    // bloco de código 1  
} else {  
    // bloco de código 2  
}
```

sintaxe:

Que significa que se a condição do if for verdadeira, o código é executado. Se não for, pulamos para o código do bloco em else.

Temos, além do if/else, uma outra maneira de criar fluxos para nossas estruturas e que nos dá mais poder de escolha, que é utilizando o else if. Isso permite incluir outras condições para comparação.

Sua estrutura é a seguinte:

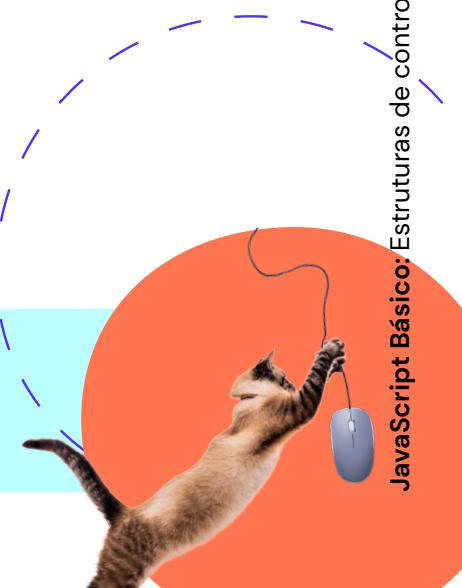
```
if (condicao1) {  
    // bloco de código 1  
} else if (condicao2) {  
    // bloco de código 2  
} else {  
    // bloco de código 3  
}
```

```
<title>
```

PULO DO GATO

```
</title>
```

- O operador ternário realiza de maneira mais reduzida uma comparação parecida com if else.





O ternário funciona com a seguinte sintaxe:

```
let variavel = condicao ? valorSeForVerdadeiro : valorSeForFalso;
```

```
<title>
```

MÃO + + NA MASSA

```
</title>
```



Crie duas variáveis e atribua a elas os seguintes valores:

- let x = 40
- let z = 30

Em seguida, escreva um código if else e teste as seguintes opções retornando a mensagem correspondente:

- Se x for maior do que y, retornar no console: "X é maior que Y"
- Se y for maior do que x, retornar no console: "Y é maior do que X"
- Se nenhuma das opções for a correta, retornar: "X e Y são iguais"

Saber não é o bastante,
devemos aplicar. Disposição não
é o suficiente, devemos fazer.

Johann Wolfgang Von Goethe



{03.}

Switch case

Podemos, sim, criar uma estrutura com vários “else if”, mas a partir do momento que essa estrutura fica mais complicada e volumosa, talvez esse não seja o melhor caminho a se tomar.

O Switch case nos dá o poder de comparar um valor com várias opções diferentes de uma maneira mais sucinta e limpa.

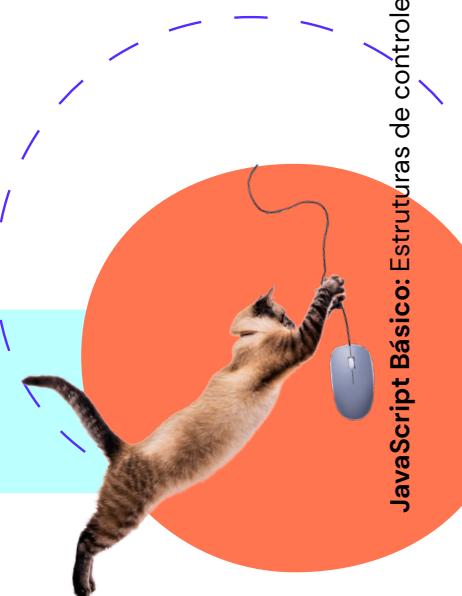
Além dessa diferença, o if/else é ótimo para condições que resultam em booleano, true or false. Enquanto o switch é indicado para valores fixos.

```
<title>
```

PULO DO GATO

```
</title>
```

- › Quando houver a necessidade de verificar **mais do que 3 condições**, prefira utilizar o **Switch case**.





<title>

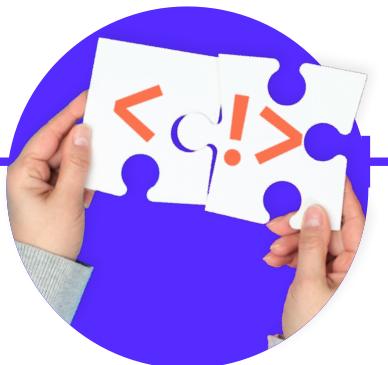
MÃO + + NA MASSA

</title>

Uma escola de ballet pretende classificar seus dançarinos de acordo com a idade, para assim dividi-los em turmas. **Crie uma estrutura switch** que utilize os seguintes dados e retorne no console o nome da respectiva turma quando forem testadas as idades:

- 8, 9, 10 - “Turma infantil”
- 11, 12, 13 - “Turma juvenil”
- 14, 15, 16 - “Turma jovem adulto”

Adicione uma mensagem default que diga “Entre em contato com a secretaria”. Para testar, crie a variável “studentAge”, atribua a ela algumas idades de sua preferência e veja que é retornado.





<title>

DESAFIO CONQUER

</title>



if (challenge)
// start

<body>

- Crie uma função que seja uma calculadora.
- Escolha entre as seguintes operações:
 - 1 - Soma (+)
 - 2 - Subtração (-)
 - 3 - Multiplicação (*)
 - 4 - Divisão real (/)
 - 5 - Divisão inteira (%)
 - 6 - Potenciação (**)
- Valores não numéricos para escolha de operação: devolver uma operação inválida e pedir novamente.
- Operação válida: pedir para o usuário inserir dois valores para serem calculados com a operação escolhida.
- Valores não numéricos para o cálculo: devolver que é um valor inválido e pedir novamente.

</body>

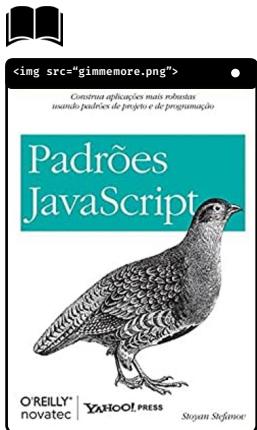
O resultado em string template:
``${n1} + ${n2} = ${resultado}``



<title>

QUERO MAIS

</title>



↗ Loops and iteration

↗ Controles de fluxo e controles de repetição



```
if (interested)
// gimmemore
```



Anotações

-
-
-
-
-
-

010101
0001001100
0100010000001010
0101000101010010
000010001001010100010
10100001010101000100010
100010100111001011100101
11110010110000101100101010100100000110010000001010