



# JavaScript [40h]

📁 Categoria	Programação
☰ Subcategoria	JavaScript
📺 Plataforma	Curso em Vídeo
👤 Canal/Instrutor	Gustavo Guanabara
📌 Status	Em andamento
📅 Início	@01/08/2024

Link da playlist do curso no Youtube:

Curso Grátis de JavaScript e ECMAScript para Iniciantes

Curso de linguagem JavaScript, voltado para iniciantes e para quem quiser aprender mais sobre ECMAScript, a versão padronizada do JS. Em um curso patrocinado...

📺 [https://www.youtube.com/playlist?list=PLHz\\_AreHm4dIsK3Nr9GVvXCbpQyHQ11o1](https://www.youtube.com/playlist?list=PLHz_AreHm4dIsK3Nr9GVvXCbpQyHQ11o1)



Meu repositório do curso:

<https://github.com/gabriellaxavier05/js-curso-em-video>

## Seções

✓ Introdução

## ▼ Módulo A: Conhecendo o JavaScript

- ☒ ~~Aula 1 — O que o JavaScript é capaz de fazer?~~
- ☒ ~~Aula 2 — Como chegamos até aqui?~~
- ☒ ~~Aula 3 — Dando os primeiros passos~~
- ☒ ~~Aula 4 — Criando o seu primeiro script~~

## ▼ Módulo B: Comandos básicos do JavaScript

- ☒ ~~Aula 5 — Variáveis e tipos primitivos~~
- ☒ ~~Aula 6 — Tratamento de dados~~
- ☒ ~~Aula 7 — Operadores (parte 1)~~
- ☒ ~~Aula 8 — Operadores (parte 2)~~

## ▼ Módulo C: Entendendo o DOM

- ☒ ~~Aula 9 — Introdução ao DOM~~
- ☒ ~~Aula 10 — Eventos DOM~~

## ▼ Módulo D: Condições em JavaScript

- ☒ ~~Aula 11 — Condições (parte 1)~~
- ☒ ~~Aula 12 — Condições (parte 2)~~
- ☒ ~~Exercícios JavaScript (parte 1)~~
- ☒ ~~Exercícios JavaScript (parte 2)~~
- ☒ ~~Exercícios JavaScript (parte 3)~~

## ▼ Módulo E: Repetições em JavaScript

- ☐ Aula 13 - Repetições (parte 1)
- ☐ Aula 14 - Repetições (parte 2)
- ☐ Exercícios JavaScript (parte 4)
- ☐ Exercícios JavaScript (parte 5)
- ☐ Exercícios JavaScript (parte 6)
- ☐ Exercícios JavaScript (parte 6)

## ▼ Módulo F: Avançando os estudos em JavaScript

- ☐ Aula 15 - Variáveis compostas

- ☐ Aula 16 - Funções
  - ☐ Exercícios JavaScript (parte 7)
  - ☐ Exercícios JavaScript (parte 8)
  - ☐ Aula 17 - Próximos passos
- 

# Anotações

## Módulo A

### ▼ Aula 01) O que o JavaScript é capaz de fazer?

É preciso entender a diferença entre ECMA e JavaScript.

Cliente x Servidor (Client x Server)

- O Servidor busca a informação no banco de dados
- O Cliente recebe do servidor a informação que procurava

O JS, inicialmente, surgiu para atender Clients, mas hoje em dia ele também atua um pouco do lado do servidor.

As interações em uma página web são feitas com JS, a parte da programação.

Ex. de estrutura web:

1. Texto e imagens ⇒ HTML
2. Design da página ⇒ CSS
3. Interações de programação ⇒ JS

JS é uma linguagem de programação.

Dentro do navegador Chrome ⇒ DevTools (aqui, se você fizer alterações, vão ser feitas só do seu lado)

### ▼ Aula 02) Como chegamos até aqui?

O JavaScript foi lançado em 1995 e, inicialmente, se chamava 'Moca'. Na época, falar "Java" estava na moda. O criador do JS, então, mudou seu antigo nome para Javascript, só pra linguagem ficar na boca do povo. Java e JS não têm nada a ver!

JS foi baseado na linguagem C.

A Netscape, dona do JS, padronizou a linguagem pela empresa Ecma, pois a Microsoft estava copiando o JS na cara dura.

\*Ecma = ISO da Europa

EcmaScript = JS padronizado

Em 2008, o Google lança o Google Chrome e em 2009 o turбина com V8. O JS rodava no navegador.

NodeJS surgiu em 2010 e é uma máquina que roda JavaScript fora do navegador (fica do lado do servido).

O EcmaScript tem várias versões. Versão atual: ES2021 (Standart).

JQuery: biblioteca JS

O Angular é do Google.

React é do Google.

Ionic: SDK para criar apps para dispositivos móveis.

## ▼ Aula 03) Dando os primeiros passos

Fontes de conteúdos sobre JS:

- Livro JavaScript: guia do programador
- Mozilla;
- Guia de referência Ecma.

Editor de código para o JS: VS Code.

Instalar o nodeJS para mexer com JS fora da máquina. Obs.: para ver se o nodeJS está funcionando após a instalação, faça uma operação simples, como  $2 + 2$ .

## ▼ Aula 04) Criando o seu primeiro script

Script interno de JS em uma página HTML:

```
<html>
  <head>
  </head>
  <body>
    <script>
      /* trecho de código JS */
    </script>
  </body>
</html>
```

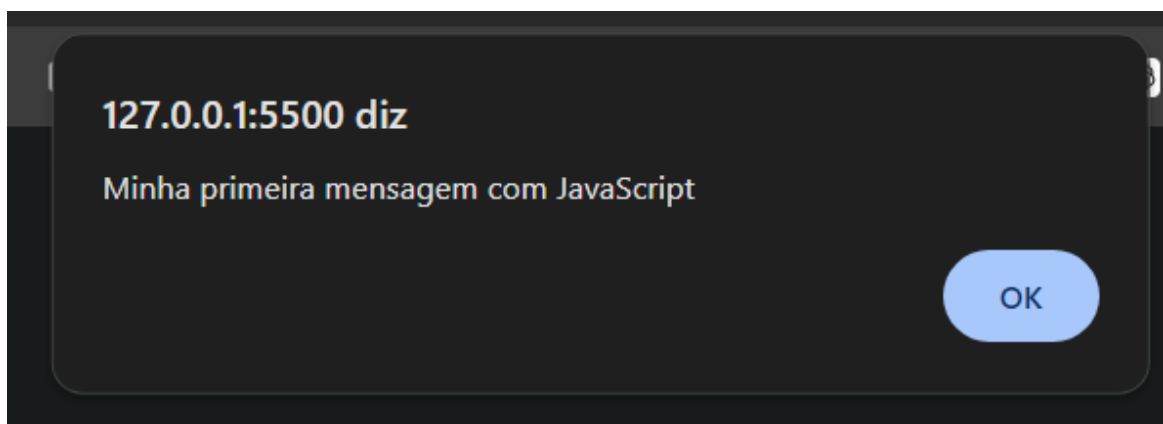
Comandos em JS geralmente são em letras minúsculas.

Não é obrigatório por ; ao final de um comando de código JS.

Criando um alerta:

```
<script>
  window.alert("Minha 1a mensagem em JS")
</script>
```

Resultado:

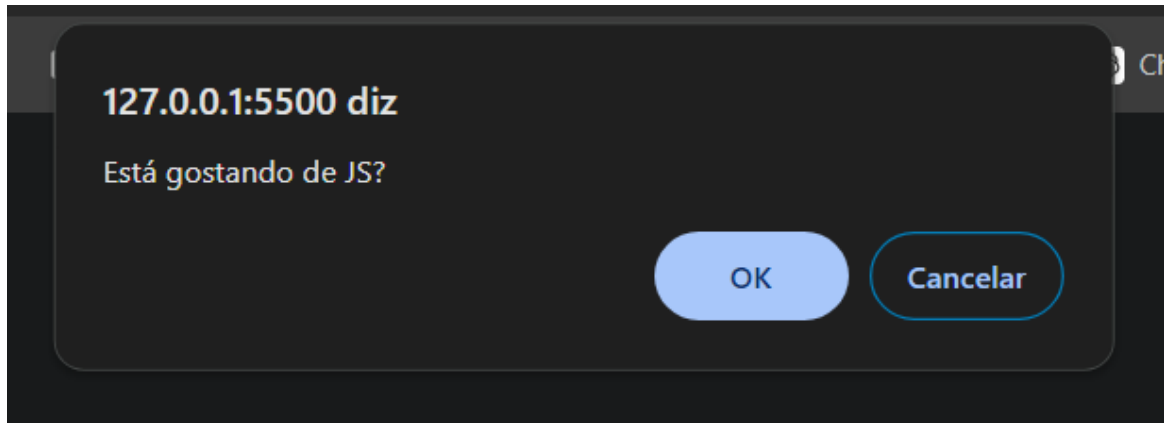


Pedindo uma confirmação (OK ou "Cancelar"):

```
<script>
  window.confirm("Está gostando de JS?")
```

```
</script>
```

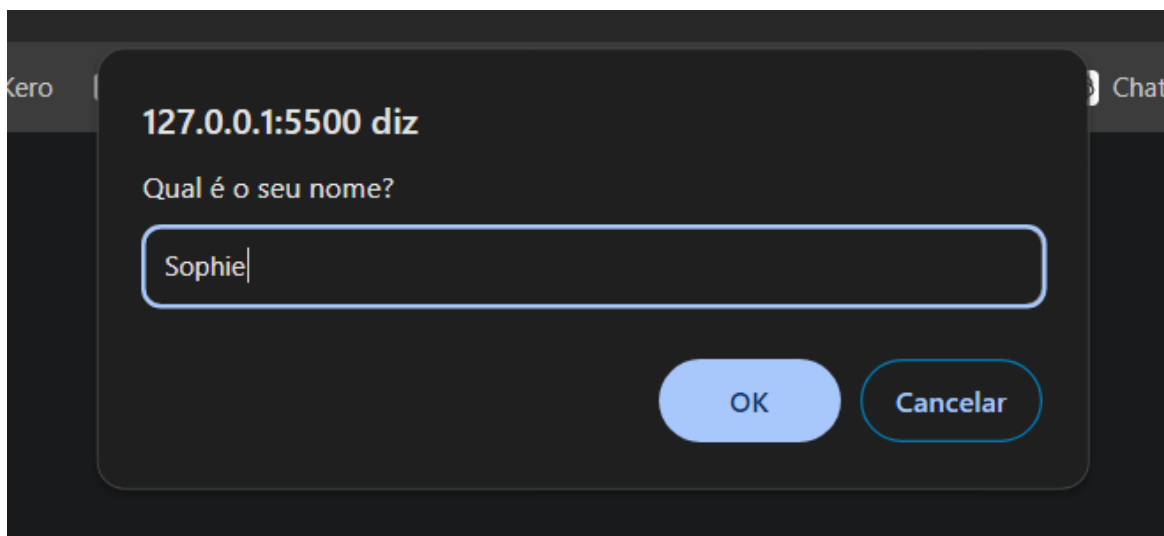
Resultado:



Pedindo uma resposta em texto ao usuário:

```
<script>  
    window.prompt("Qual é o seu nome?")  
</script>
```

Resultado:



## Módulo B

### ▼ Aula 05) Variáveis e Tipos Primitivos

Comentários em JS:

```
// comentário de 1 linha

/*
Comentário de + de 1 linha
:D
*/
```

## Variáveis em JS

Variáveis armazenam dados.

```
// exemplo
vaga a1 = carro1
// em JS, 1 símbolo de = significa "recebe"
```

Deixando uma variável nula:

```
vaga a1 = null
```

Variáveis precisam de nomes para serem identificadoras.

Em JS, dá pra usar 3 tipos de aspas para uma variável do tipo String, mas há diferenças entre elas:

- Aspas duplas: `var s1 = "JavaScript"`
- Aspas simples: `'Curso em Vídeo'`
- Crase: ``Guanabara``

Regras para variáveis:

- Podem começar com letras, \$ ou \_
- Não podem começar com números
- Dá pra usar letras e números
- Dá pra usar acentos e símbolos
- Não podem conter espaços
- Não podem ser palavras reservadas; ou seja, já usadas pelo JS para algum comando (ex.: function, alert)

Obs.: use o nodeJS para praticar exercícios com variáveis.

Para exibir o valor de uma variável no nodeJS, digite o nome da variável em questão. Ex.:

```
var nome = "Gabriella"
nome
// resultado: "Gabriella"
```

`ctrl + shift + `` ⇒ abre o terminal no VS Code.

Para entrar no nodeJS pelo VS Code no terminal: digitar `node`

Tipos de dados em JS:

- Number: 5, -12, 0.5, 15.9, 3.14
  - Infinity
  - NaN
- Booleano: true or false
- String: Cadeia de caracteres. Ex.: `"Google"`, `'JavaScript'`, ``Maria``
- Null
- Undefined
- Object:



- Array
- Function

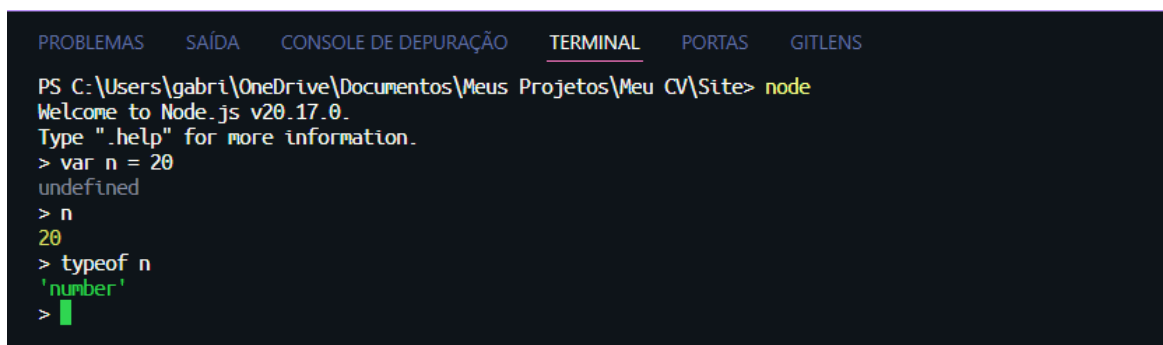
Para saber o tipo de uma variável: `typeof`

Ex.:

```
var n = 20
n // pra exibir o resultado
typeof n // pra saber o tipo de uma variável
```

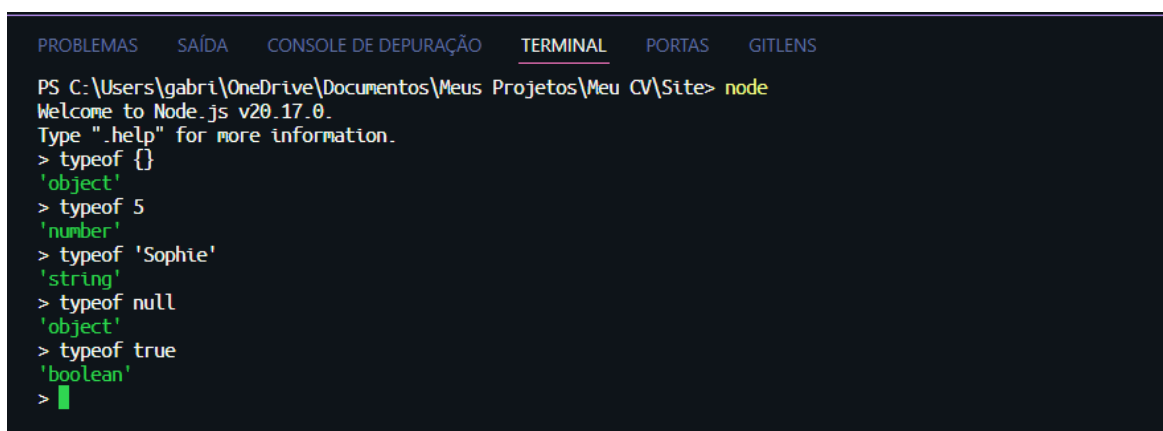
Obs.: se você digitar `typeof` número, `{}`, nome ou `true/false`, o nodeJS vai identificar o tipo de dado pra você.

Ex.:



```
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  PORTAS  GITLENS

PS C:\Users\gabri\OneDrive\Documentos\Meus Projetos\Meu CV\Site> node
Welcome to Node.js v20.17.0.
Type ".help" for more information.
> var n = 20
undefined
> n
20
> typeof n
'number'
> █
```



```
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  PORTAS  GITLENS

PS C:\Users\gabri\OneDrive\Documentos\Meus Projetos\Meu CV\Site> node
Welcome to Node.js v20.17.0.
Type ".help" for more information.
> typeof {}
'object'
> typeof 5
'number'
> typeof 'Sophie'
'string'
> typeof null
'object'
> typeof true
'boolean'
> █
```

Pra fechar o nodeJS no terminal do VS Code: `.exit`

Pra fechar o terminal do VS Code: `exit`

Aula e exercício relacionado ao conteúdo: pasta "aula04" e "ex01"

## ▼ Aula 06) Tratamento de dados

Pasta de exercícios do conteúdo/aula em questão: aula06

Array é uma variável especial.

Tipos de dados primitivos + importantes em JS:

- Number
- String

Para começar a manipular dados com variáveis em JS, vamos usar como exemplo o comando `window.prompt`.

### Trabalhando com variáveis do tipo String

No código HTML (JS interno):

```
<script>
  var nome = window.prompt('Qual é o seu nome?')
  window.alert('Prazer em conhecer, ' + nome + '!')
  // O código acima pergunta ao usuário o seu nome, ar
  mazen a resposta na variável 'nome' e a exibe em um ale
  rta. O + aqui serve como concatenação (e só funciona ass
  im porque as informações antes e depois do + são do tipo
  String)
</script>
```

### Trabalhando com variáveis do tipo Number

Obs.: o comando `window.prompt()` trata as variáveis dentro dele com String. Para trabalhar com number, é necessário fazer a conversão de dados. Se não converter, no momento da apresentação do resultado o + (que serve como concatenação no `window.alert()`) vai servir como um operador de soma, e tudo ficará bugado.

## Conversão de String para Number em JS:

- Int (Inteiros): `Number.parseInt(variavel)`
- Float (Real): `Number.parseFloat(variável)`

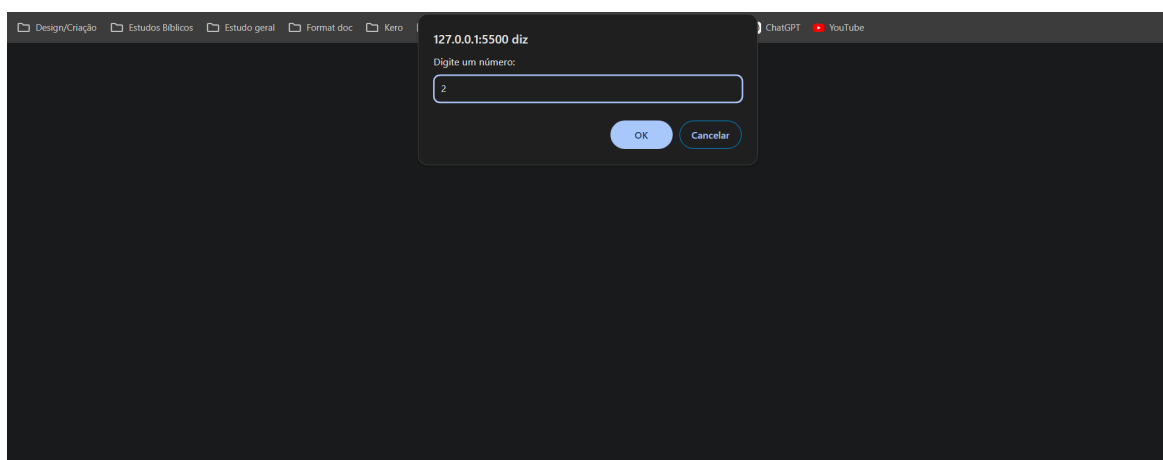
Ex. de código:

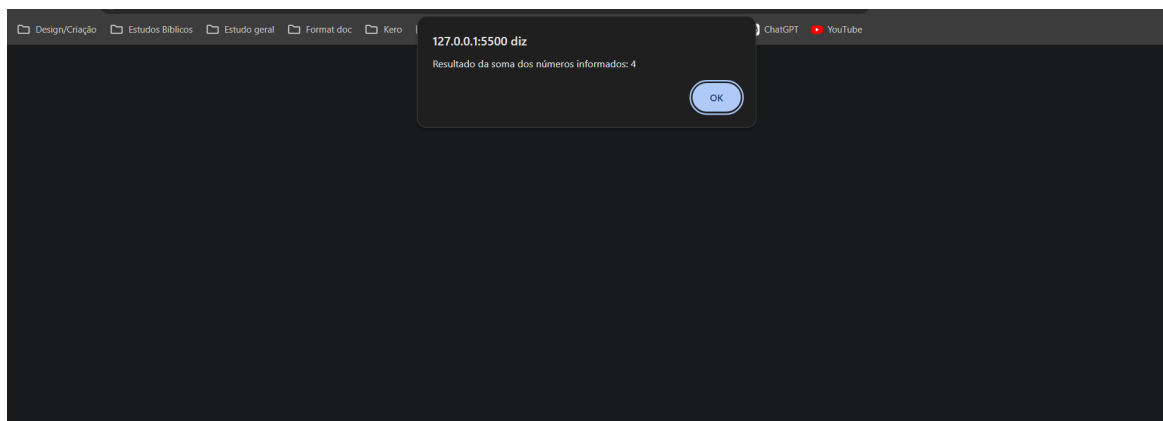
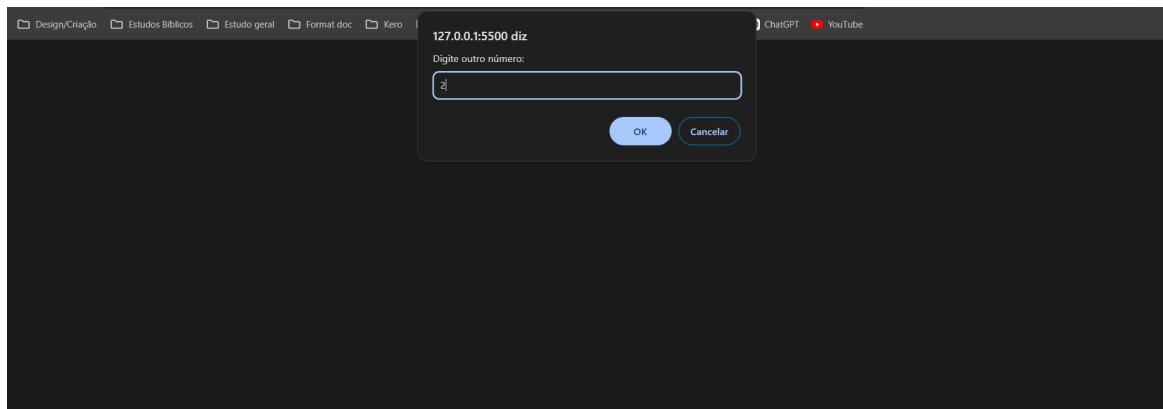
```
<script>
    // lendo 2 valores informados pelo usuário e convertendo-os em números inteiros para que a exibição da soma ocorra sem problemas
    var n1 = Number.parseInt(window.prompt('Digite um número: '))
    var n2 = Number.parseInt(window.prompt('Digite o outro número: '))

    // somando os números informados pelo usuário:
    var s = n1 + n2

    // apresentando o resultado:
    window.alert('Resultado da soma dos números informados: ' + s)
</script>
```

Apresentação:





Dá pra simplificar as conversões de String para Number apenas usando `Number(variável)` nas versões mais recentes do JS. Ex.:

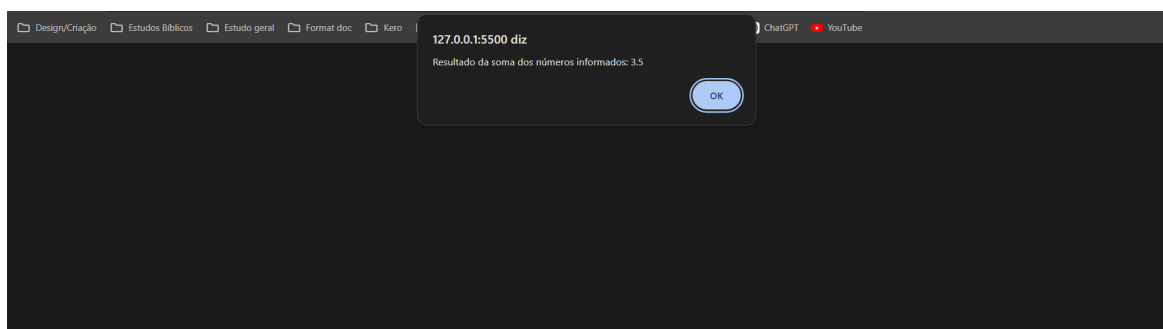
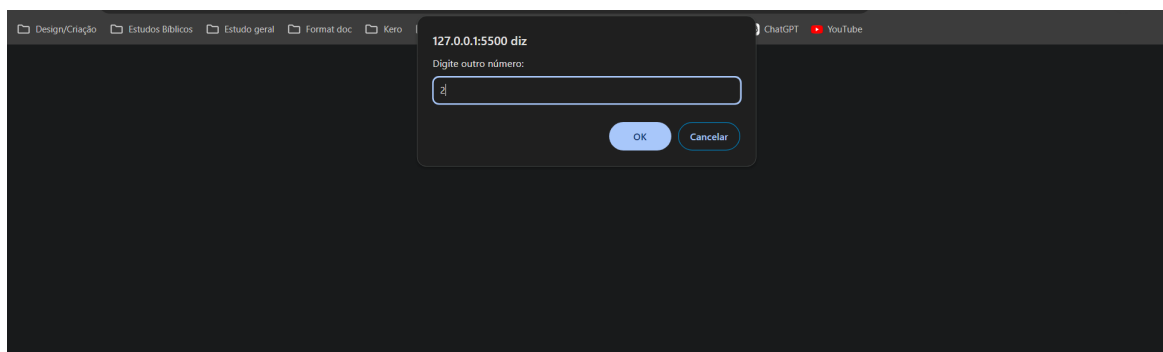
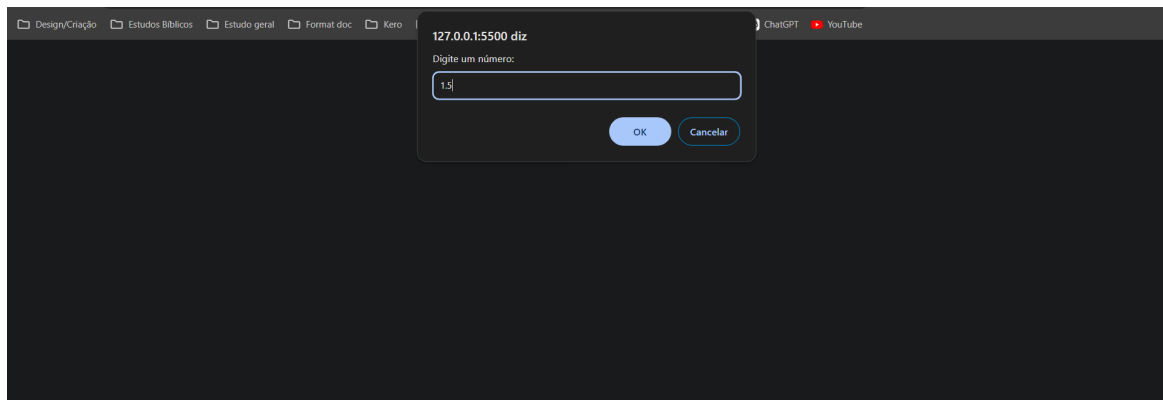
Código:

```
// Usando somente Number() para deixar o JS decidir o tipo de dado:
var n1 = Number(window.prompt('Digite um número: '))
var n2 = Number(window.prompt('Digite outro número: '))

// somando os números informados pelo usuário:
var s = n1 + n2

// apresentando o resultado:
window.alert('Resultado da soma dos números informados: ' + s)
```

Apresentação:



Obs.: para informar números reais em JS, usa-se `1.5`, não `1.5`! Ex.: 0.1, 7.5

## Convertendo dados do tipo Number para String

- `String(variável)`
- `variável.toString()`

## Outras conversões

```
Number(nomeV) // converte para Number
parseInt(nomeV) // converte para int
```

```
parseFloat(nomeV) // converte para float
String(nomeV) // converte para String
```

## Formatando variáveis

```
var s = 'JavaScript'
'Eu estou estudando s' // não faz interpolação (não apresenta o valor de s)
'Eu estou estudando ' + s // usa concatenação
`Eu estou estudando ${s}` // usa template string. Lembre-se o uso da crase é OBRIGATÓRIO!
```

Trabalhando com o nodeJS (apresentação do exemplo acima):



```
PS C:\Users\gabri\OneDrive\Documentos\Estudos\cursos\Curso em Vídeo\JavaScript\aula06> node
Welcome to Node.js v20.17.0
Type ".help" for more information.
> var s = 'JavaScript'
undefined
> s
'JavaScript'
> 'Eu estou estudando s'
'Eu estou estudando s'
> 'Eu estou estudando ' + s
'Eu estou estudando JavaScript'
> `Eu estou estudando ${s}`
'Eu estou estudando JavaScript'
```

Ex. 2:

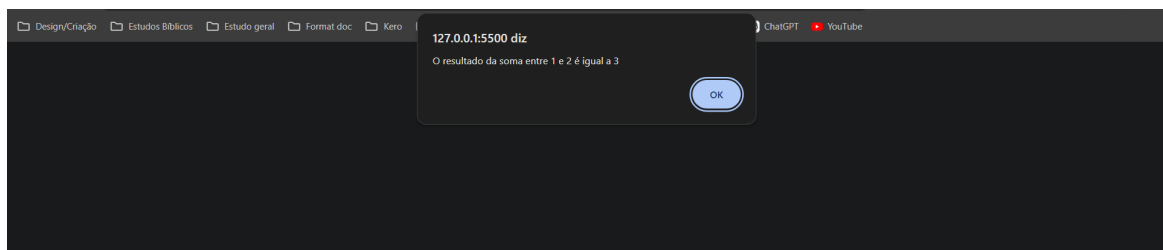
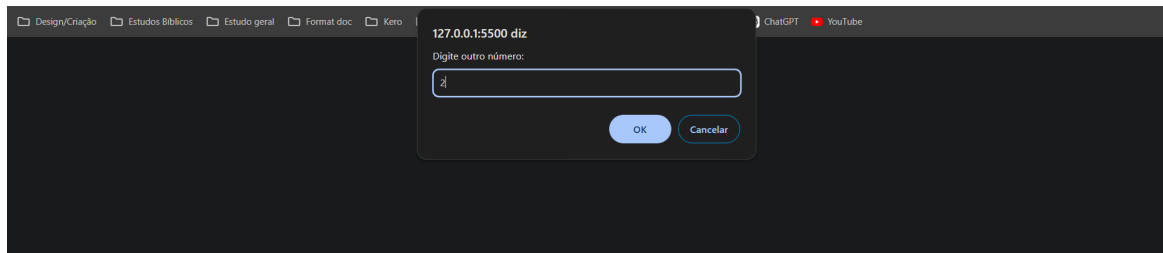
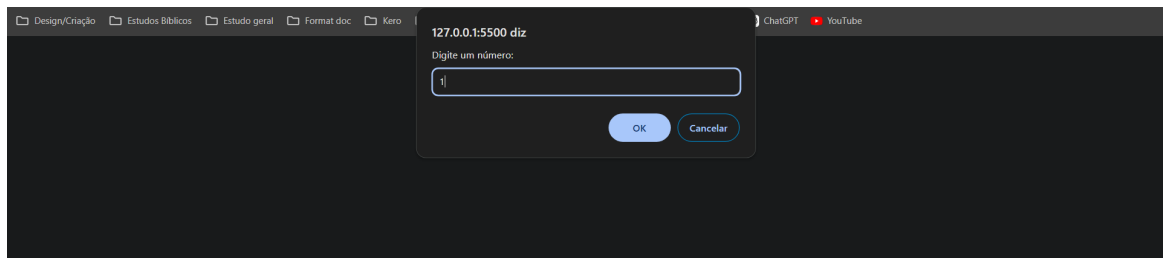
Código:

```
// Usando somente Number() para deixar o JS decidir o tipo de dado:
var n1 = Number(window.prompt('Digite um número: '))
var n2 = Number(window.prompt('Digite outro número: '))

// somando os números informados pelo usuário:
var s = n1 + n2

// apresentando o resultado da soma com template string:
window.alert(`O resultado da soma entre ${n1} e ${n2} é igual a ${s}`)
```

Apresentação:



Outras coisas para serem feitas com String:

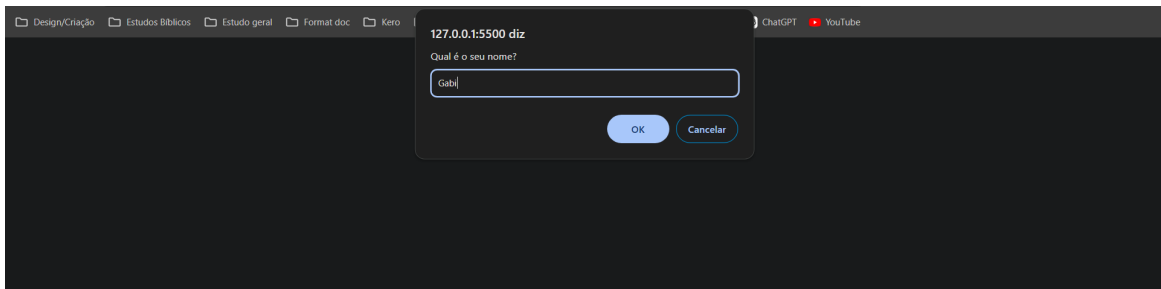
```
var s = 'JavaScript'
s.length() // quantos caracteres a String tem
s.toUpperCase() // tudo para MAIÚSCULO
s.toLowerCase() // tudo para MINÚSCULO
```

Usando as funções mencionadas acima:

```
var nome = window.prompt('Qual é o seu nome?')
document.write(`<h2>Seu nome tem ${nome.length} letras.
</h2>`) // document.write = escreva no documento (dentro
do corpo da página HTML mesmo)
document.write(`<br> Seu nome em maiúsculo é ${nome.toUp
perCase()}`) // apresenta o nome informado em letras mai
úsculas
document.write(`<br> Seu nome em minúsculo é ${nome.toLo
```

```
werCase()}}` ) // apresenta o nome informado em letras min  
úsculas
```

Resultado:



## Exerício 04 - Mód B - Aula 06

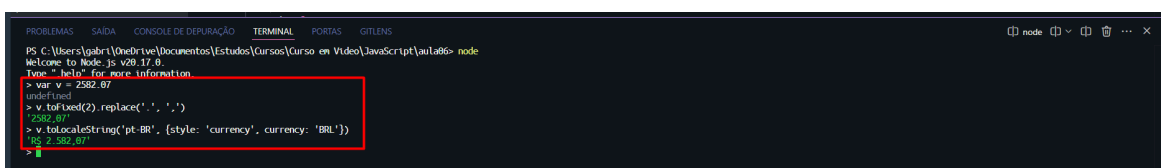
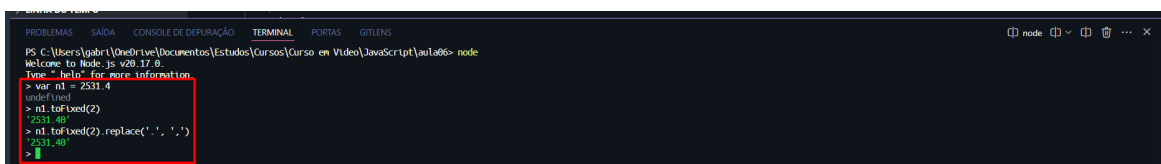
Seu nome tem 4 letras.

Seu nome em maiúsculo é GABI  
Seu nome em minúsculo é gabi

## Formatando números com JS

- `variável.toFixed(2)` → acrescenta 2 casas após a vírgula (para números decimais)
- `variável.toFixed(2).replace('.', ',')` → acrescenta 2 casas após a vírgula (para números decimais e troca o "." por ",").
- `variável.toLocaleString('pt-BR', {style: 'currency', currency: 'BRL'})` → define o valor da variável no padrão da moeda da região especificada

Exemplo:





## ▼ Aula 07) Operadores (parte 1)

O JS possui vários operadores. Alguns deles:

- Aritméticos
- Atribuição
- Relacionais
- Lógicos
- Ternário

### Operadores aritméticos

Operador	Significado
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Divisão inteira. Pega o resto da divisão
**	Potência

Cuidado ao usar operadores aritméticos, pois há uma ordem de resolução ao usá-los! Ex.:

$$5 + 3/2$$

Acima (👉), O JS entende que primeiramente a resolução não é da soma, mas sim da divisão. Então fica:

1.  $3/2 = 1,5$

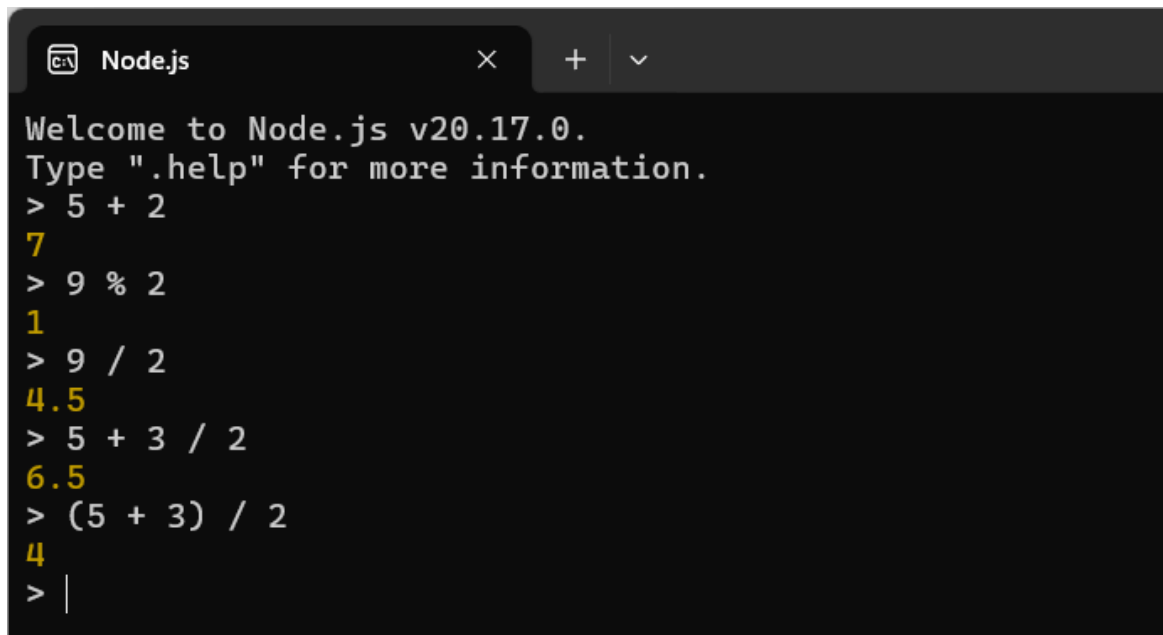
2.  $5 + 1,5 = 6,5$

Para querer que  $5 + 3$  sejam resolvidos primeiro e juntos, é necessário colocá-los em parênteses. Veja abaixo:

$$(5 + 3)/2$$

$$8/2 = 4$$

Operações no Node.js:



```
Node.js
Welcome to Node.js v20.17.0.
Type ".help" for more information.
> 5 + 2
7
> 9 % 2
1
> 9 / 2
4.5
> 5 + 3 / 2
6.5
> (5 + 3) / 2
4
> |
```

Ordem de precedência de operadores no JS:

1. `( )`
2. `**`
3. `* / %`
4. `+ -`

## Atribuição simples

Exemplo:

```
var a = 5 + 3 // a RECEBE 5 + 3; = 8
var b = a % 5 // b RECEBE o resto de a dividido por 5; = 3
var c = 5 * b ** 2 // para resolver essa equação normalm
ente, seria tipo 5 8 (b ** 2); = 45
var d = 10 - a / 2 // organização para resolução: 10 -
(a / 20); = 6
```

```
var e = 6 * 2 / d // organização para resolução: (6 * 2)
/ d; = 2
var f = b % e + 4 / e // organização para resolução: (b
% e) + (4 / e); = 2
```

## Auto-atribuições

Exemplos:

```
var n = 3 // n é igual a 3
var n = n + 4 // n é igual a n + 4; n = 3 + 4 => 7
n = n - 5 // n é igual a n - 5; n = 7 - 5 => 2
n = n * 4 // n é igual a n vezes 4; n = 2 * 4 => 8
n = n / 2 // n é igual a n dividido por 2; n = 8 / 2 =>
4
n = n ** 2 // n é igual a n ao quadrado; n = 4 ** 2 => 1
6
n = n % 5 // n é igual ao resto de n dividido por 5; n =
16 % 5 => 1
```

Simplificando as atribuições:

```
n += 4 // n = n + 4
n -= 5 // n = n - 5
n *= 4 // n = n * 4
n /= 2 // n = n / 2
n **= 2 // n = n ** 2
n %= 5 // n = n % 5
```

## Incremento e decremento

Exemplos:

```
var x = 5
x = x + 1 // 5 = 5 + 1 => 6
```

```
x = x - 1 // 6 = 6 - 1 => 5
```

```
x ++ // = x = x + 1
```

```
x -- // x = x - 1
```

No Node.js:

```
Node.js
> var n = 10
undefined
> n
10
> n++
10
> n
11
> n--
11
> n
10
> |
```

## ▼ Aula 08) Operadores (parte 2)

Operadores de relação em JS:

Operador	Significado
>	Maior
<	Menor
>=	Maior ou igual
<=	Menor ou igual
==	Igual
!=	Diferente

O resultado de expressões usando operadores sempre será booleano — ou seja, verdadeiro ou falso. Ex.: 5 > 2 | TRUE

Ao usar operadores relacionais e operadores aritméticos na mesma linha, primeiramente os aritméticos são resolvidos, depois os relacionais.

Ex. no NodeJS:

```
Node.js
> 5 > 2
true
> 8 < 4
false
> var a = 8
undefined
> var b = 15
undefined
> a > b
false
> a < b
true
> a <= b - 10
false
> |
```

Igualdade no JS: `==`

## Identidade

O sinal de igualdade NÃO testa o tipo, no JS. Veja a explicação:

```
5 == 5 // resultado = true
5 == '5' // resultado = true. Isso porque o JS NÃO testa o tipo do que foi analisado. Por lógica, '5' não seria considerado igual a 5 por estar dentro das aspas (String)
5 === '5' // resultado = false. Os === verificarão se os elementos comparados têm o mesmo valor e tipo
```

Ex. 2 no NodeJS:

```
Node.js
Welcome to Node.js v20.17.0.
Type ".help" for more information.
> 5 == 5
true
> 5 == '5'
true
> 5 === '5'
false
> |
```

## Operadores lógicos

Operador	Significado
!	Negação
&&	Conjunção (E)
	Disjunção (OU)

### Negação ( ! )

Opção 1	Operador	Opção 2	Resultado
false	!	true	false
true	!	false	true

### Conjunção

Opção 1	Operador	Opção 2	Resultado
true	&&	true	true
true	&&	false	false
false	&&	true	false
false	&&	false	false

Só serei satisfeita se as duas opções forem verdadeiras.

## Disjunção

Opção 1	Operador	Opção 2	Resultado
true		true	true
true		false	true
false		true	true
false		false	false

Basta um deles ser verdadeiro, que o resultado será verdadeiro!

Exemplo no NodeJS:

```
Node.js
> var a = 5
undefined
> var b = 8
undefined
> true && false
false
> false || false
false
>
> a > b && b % 2 == 0
false
> |
```

Ordem de resolução dos operadores lógicos:

1. `!`
2. `&&`
3. `||`

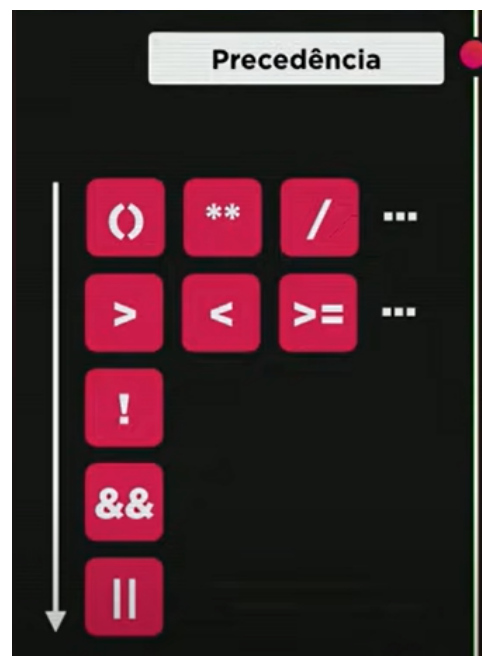
Lembrando novamente que: primeiros é resolvido o uso dos operadores aritméticos, depois os relacionais e depois os lógicos!

Exemplos:

```
// Exemplos

idade >= 15 && idade <= 17 // a idade está entre 15 e 17?
estado == 'RJ' || estado == 'SP' // o estado é RJ ou SP?
salário > 1500 && sexo != 'M' // o salário é acima de 1500 e não é homem?
```

Relembrando a precedência:



## Operador ternário

Operadores: `?`, `:`

Se chamam ternários por conta de sua estrutura (junta 3 operandos): teste

`?` true `:` false

Entendendo:

- 1o operando: teste lógico. Dá verdadeiro ou falso.
- 2o operando (o que está no meio): o que vai acontecer quando o teste lógico for verdadeiro.



- 3o operando: o que vai acontecer quando o teste lógico for falso.

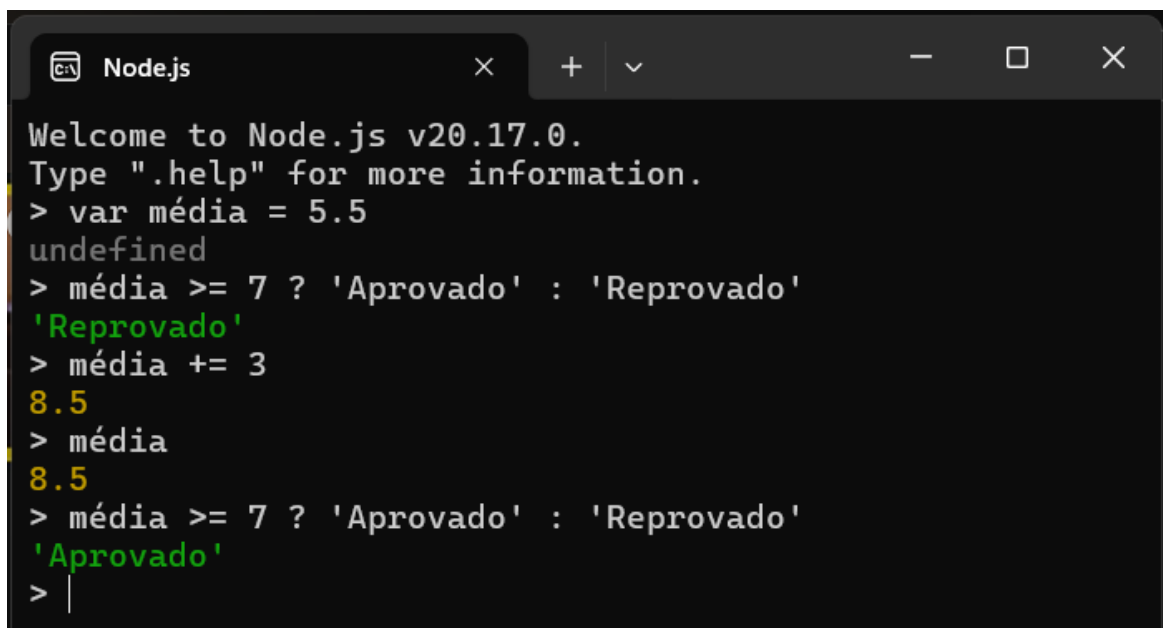
Ex.:

```
média >= 7.0 ? "Aprovado" : "Reprovado"
```

Significado da expressão acima: Se a média for maior ou igual a 7, resultado: "Aprovado". Senão, "Reprovado".

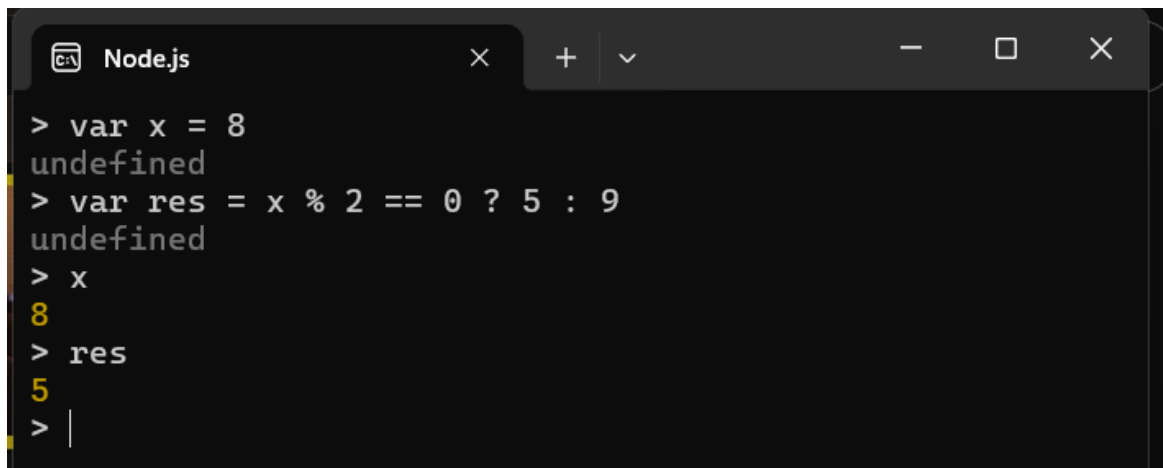
No NodeJS:

Ex. 1: Verificando a média de um aluno



```
Node.js
Welcome to Node.js v20.17.0.
Type ".help" for more information.
> var média = 5.5
undefined
> média >= 7 ? 'Aprovado' : 'Reprovado'
'Reprovado'
> média += 3
8.5
> média
8.5
> média >= 7 ? 'Aprovado' : 'Reprovado'
'Aprovado'
> |
```

Ex. 2: O resto de x / 2 será igual a 5 ou a 9?



```
Node.js
> var x = 8
undefined
> var res = x % 2 == 0 ? 5 : 9
undefined
> x
8
> res
5
> |
```

## Diferenças entre `=`, `==` e `===`

Operador	Significado
<code>=</code>	Atribuição
<code>==</code>	Comparação do valor
<code>===</code>	Comparação do valor e do tipo de dado

## Módulo C: Entendo o DOM

### ▼ Aula 09) Introdução ao DOM

Pasta de exercícios relacionados: aula09

Extensões para instalar no VS Code:

- Live Server (abre a página HTML que você estiver trabalhando)
- Node.js Exec (habilita a tecla F8 para rodar o JS pelo Node)

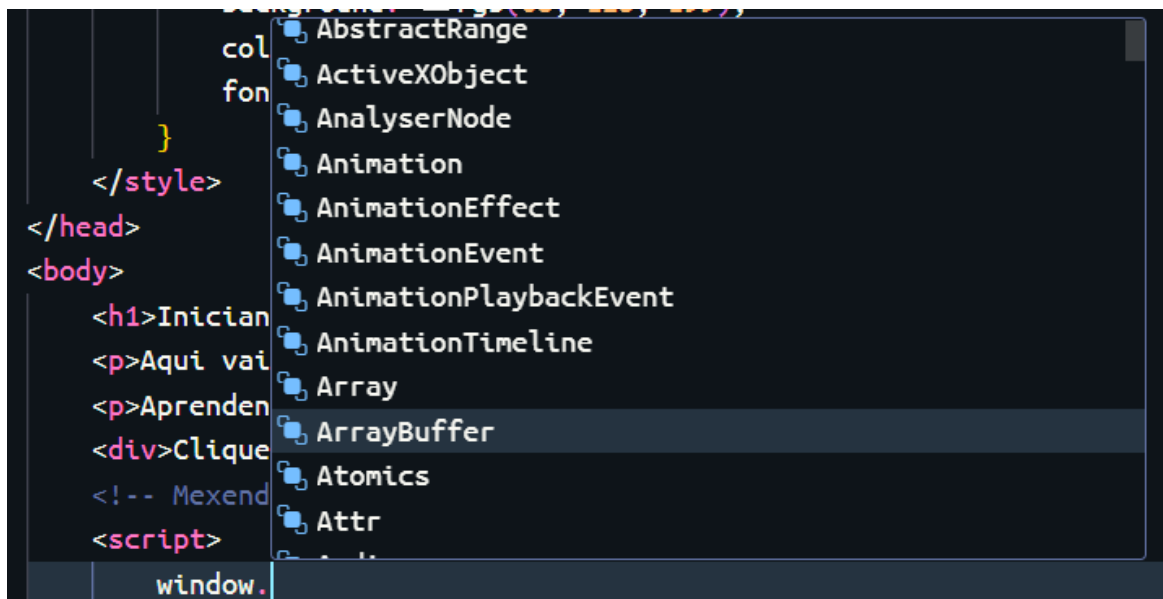
DOM = Document Object Model. É um modelo de objetos para documentos. Dá aos navegadores acesso aos componentes internos do seu site. Não funciona dentro do Node.js.

Há uma árvore DOM, que é formada da seguinte forma:

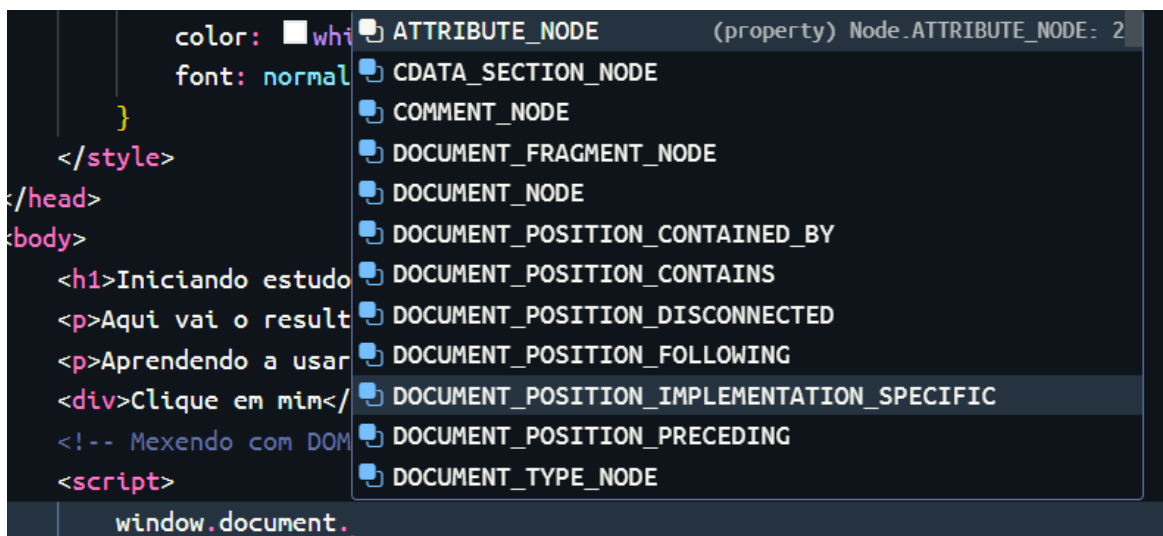
- window (janela);
  - location (diz a localização do seu site; qual é a URL, qual é página atual, qual foi a página anterior...)
  - document
    - html
      - head
        - meta
        - title
      - body

- h1
- p
  - strong
- div
- history (guarda de onde você veio e para onde você vai)

Elementos dentro de `window`:



Elementos dentro de `document`:

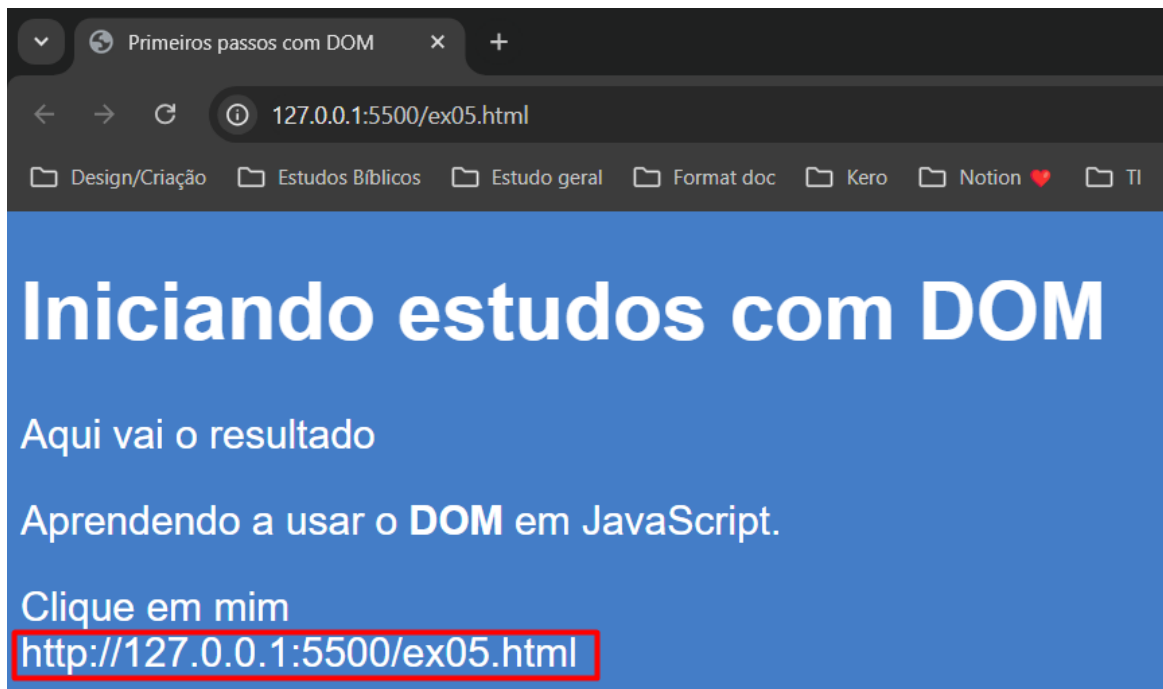


Ex.:

Código:

```
21 <script>
22   window.document.write(window.document.URL) //puxa a URL do documento
23 </script>
```

Resultado:



Técnicas de acesso aos elementos do JS:

- Por marca:

```
getElementsByTagName()
```

- Por ID:

```
getElementById()
```

- Por nome

```
getElementsByName()
```

- Por classe

```
getElementsbyClassName()
```

- Por seletor

```
querySelector()
```

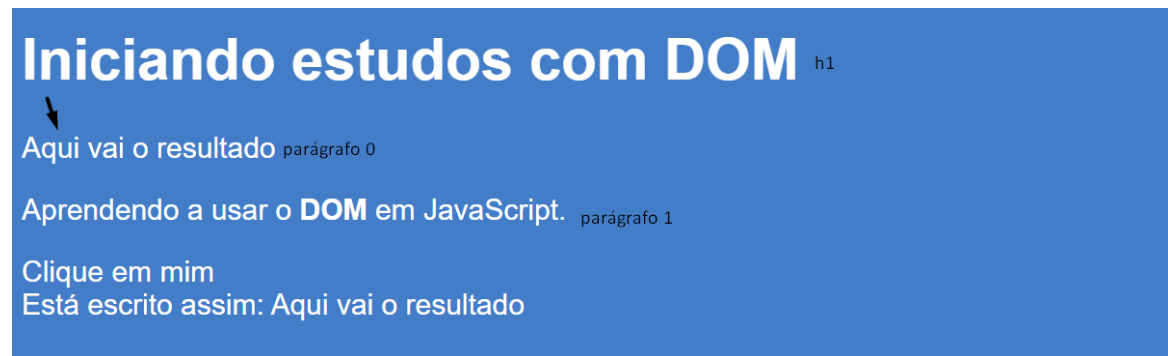
## Selecionando elementos

Ex.:

Código:

```
24 // fazendo a seleção de elementos
25 var p1 = window.document.getElementsByTagName('p')[0] // seleciona parágrafos (no caso, vai selecionar o parágrafo zero)
26 window.document.write('Está escrito assim: ' + p1.innerText) // escreve na tela o texto do 1o parágrafo
```

Resultado:



Obs.:

- InnerHTML: pego o trecho de código HTML formatado.
- InnerText: pega o trecho de código HTML sem formatação.


Exemplos:

Com InnerHTML:

```
15 <body>
16   <h1>Iniciando estudos com DOM</h1>
17   <p>Aqui vai o resultado</p>
18   <p>Aprendendo a usar o <strong>DOM</strong> em JavaScript.</p> Parágrafo 1
19   <div>Clique em mim</div>
20   <!-- Mexendo com DOM -->
21   <script>
22     /* window.document.write(window.document.URL) */ //puxa a URL do documento
23
24     // fazendo a seleção de elementos
25     var p1 = window.document.getElementsByTagName('p')[1] // seleciona parágrafos (no caso, vai selecionar o parágrafo 1)
26     /* window.document.write('Está escrito assim: ' + p1.innerText) */ // escreve na tela o texto do 1o parágrafo
27
28     window.document.write(p1.innerHTML) // escreve o parágrafo 1 exatamente com a mesma formatação HTML
29   </script>
30 </body>
```

# Iniciando estudos com DOM

Aqui vai o resultado

Aprendendo a usar o **DOM** em JavaScript.  Parágrafo 1

Clique em mim


Aprendendo a usar o **DOM** em JavaScript.

Com InnerText:

```
15 <body>
16   <h1>Iniciando estudos com DOM</h1>
17   <p>Aqui vai o resultado</p>
18   <p>Aprendendo a usar o <strong>DOM</strong> em JavaScript.</p>
19   <div>Clique em mim</div>
20   <!-- Mexendo com DOM -->
21   <script>
22     /* window.document.write(window.document.URL) */ //puxa a URL do documento
23
24     // fazendo a seleção de elementos
25     var p1 = window.document.getElementsByTagName('p')[1] // seleciona parágrafos (no caso, vai selecionar o parágrafo 1)
26     /* window.document.write('Está escrito assim: ' + p1.innerText) */ // escreve na tela o texto do 1o parágrafo
27
28     /* window.document.write(p1.innerHTML) */ // escreve o parágrafo 1 exatamente com a mesma formatação HTML
29     window.document.write(p1.innerText) // escreve o parágrafo 1 sem a formatação HTML
30   </script>
31 </body>
```

# Iniciando estudos com DOM

Aqui vai o resultado

Aprendendo a usar o **DOM** em JavaScript.  Parágrafo 1

Clique em mim

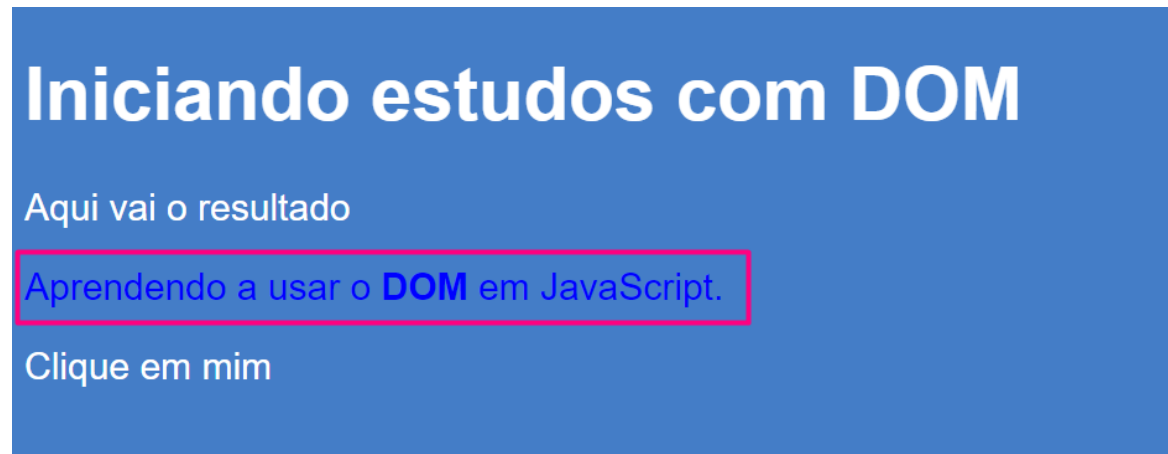
Aprendendo a usar o **DOM** em JavaScript.

## Mudando a cor de um parágrafo com JS

Código:

```
31 // Mudando a cor de um parágrafo
32 p1.style.color = 'blue'
33 </script>
```

Resultado:

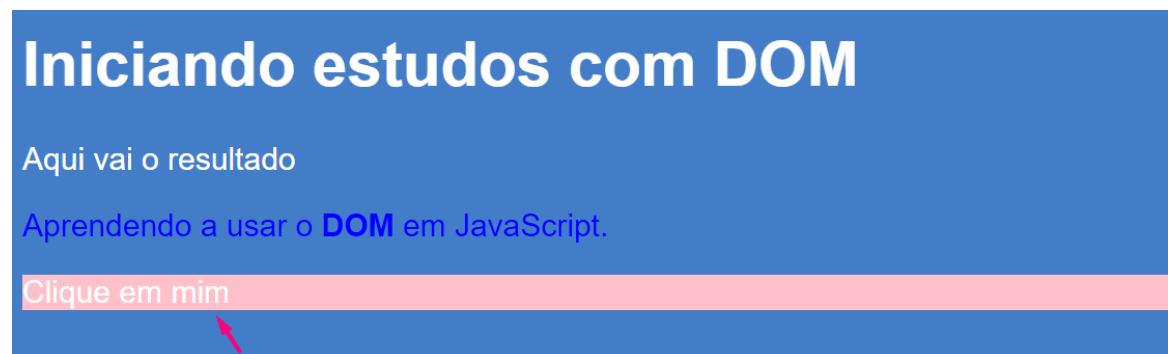


**Mexendo com** `getElementById` :

Código:

```
34 // Mexendo com getElementById
35 var d = window.document.getElementById('msg')
36 d.style.background = 'pink'
```

Resultado:



**Mexendo com** `getElementByName` :

```
<body>
  <div name="msg"> Texto </div>
  <script>
    var d = window.document.getElementsByName('msg')
```

```
[0]
</script>
</body>
```

**Mexendo com** `getElementByClassName` :

```
<body>
  <div class="msg"> Texto </div>
  <script>
    var d = window.document.getElementsByClassName
    ('msg')[0]
  </script>
</body>
```

**Mexendo com seletores:**

QuerySelector: serve para "economizar" linhas de comando

```
19      <div id="msg">Clique em mim</div>
```

```
40      // Usando querySelector
41      var d = window.document.querySelector('div#msg')
42      d.style.background = 'magenta'
```

## Iniciando estudos com DOM

Aqui vai o resultado

Aprendendo a usar o **DOM** em JavaScript.

Clique em mim

LEMBRANDO QUE (Nada a ver com JS, mas sim com HTML):

- Id → .



- Class → #

## ▼ Aula 10) Eventos DOM

Pasta de exercícios relacionada: aula10

Com o DOM, acessamos os elementos da nossa página com JavaScript.

Evento é tudo o que pode acontecer com um elemento HTML.

Tipos de eventos:

- Mouse:
  - mouseenter
  - mousedown
  - click
  - mousemove
  - mouseup
  - mouseout

Para que um evento seja disparado, é necessário usar funções.

Função: conjunto de códigos/linhas que só serem executadas quando um evento ocorrer. Toda função funciona dentro de um bloco, que é representado por `{ }`. Ex. de bloco:

```
function {  
  
    // bloco função anônima  
  
}
```

No JS, função que não tem nome é chamada de anônima e é permitido. Mas para que o método possa funcionar, é necessário dar um título à função.

```
function ação(parâmetros){  
  
    // bloco função com nome
```

```
}
```

Os eventos podem ser disparados na parte do JS ou no HTML. Depende de como você preferir.

Ex. de disparos de eventos:

Código:

```
19 <body>
20   <!-- disparando um evento com um método dentro do elemento HTML-->
21   <div id="area" onclick="clicar()"> <!-- Quando eu clicar em cima da div... -->
22     <p>Interaja...</p>
23   </div>
24
25   <!-- JS -->
26   <script>
27     // Criando o bloco
28     function clicar() {
29       // Configurando o método clicar() para o que acontecerá quando a ação for executada
30       var area = window.document.getElementById('area')
31       area.innerText = 'Clicou!'
32       // O que acontece quando o usuário clicar na div: o texto da div "area" mudará para "Clicou!" quando for clicada.
33     }
34   </script>
35 </body>
```

Resultado:

Antes:



Depois:



Exemplo 2:

Código:

```
20 <!-- disparando eventos com métodos dentro do elemento HTML-->
21 <div id="area" onclick="clicar()" onmouseenter="entrar()" onmouseleave="sair()"> <!-- Quando eu executar essas ações... -->
22 <p>Interaja...</p>
23 </div>
24
25 <!-- JS -->
26 <script>
27     // Criando o bloco
28
29 > /* Modelo 1 - 1o exemplo...
37
38     /* Modelo 2 - 2o exemplo */
39     var area = window.document.getElementById('area') // Estando do "lado de fora" serve para todos os outros métodos
40     function clicar(){
41         area.innerText = 'Clicou!'
42     }
43
44     function entrar(){
45         area.innerText = 'Entrou!'
46     }
47
48     function sair(){
49         area.innerText = 'Saiu!'
50     }
51 </script>
52 </body>
```

Execução:

Ao passar o mouse em cima da div:



Ao retirar o mouse de cima da div:



Ao clicar na div:



Pra mudar a cor de fundo de um elemento HTML ao executar alguma ação:

```
// Dentro da função:  
nomeElemento.style.background = 'cor'
```

Disparando eventos pelo JS (os do exemplo acima foram pelo HTML):

```
22 <div id="area"> <!-- Quando eu executar essas ações... -->  
23 <p>Interaja...</p>  
24 </div>  
25  
26 <!-- JS -->  
27 <script>  
28 // Criando o bloco  
29  
30 > /* Modelo 1 - 1o exemplo...  
31  
32  
33  
34  
35  
36  
37  
38  
39 /* Modelo 2 - 2o exemplo */  
40 var area = window.document.getElementById('area') // Estando do "lado de fora" serve para todos os outros métodos  
41 /* Disparando eventos pelo próprio JS */  
42 area.addEventListener('click', clicar) // quando a div for clicada, o método clicar() será chamado  
43 area.addEventListener('mouseenter', entrar) // quando o mouse parar em cima da div, o método entrar() será chamado  
44 area.addEventListener('mouseout', sair) // quando o mouse sair de cima da div, o método sair() será chamado  
45  
46 function clicar(){  
47   area.innerText = 'Clicou!' // Muda o texto da div ao clicar nela  
48   area.style.background = 'red' // Muda o cor do fundo da div ao clicar nela  
49 }  
50  
51 function entrar(){  
52   area.innerText = 'Entrou!' // Muda o texto da div ao passar o mouse em cima dela  
53 }  
54  
55 function sair(){  
56   area.innerText = 'Saiu!' // Muda o texto da div ao tirar o mouse de cima dela  
57 }  
58 </script>
```

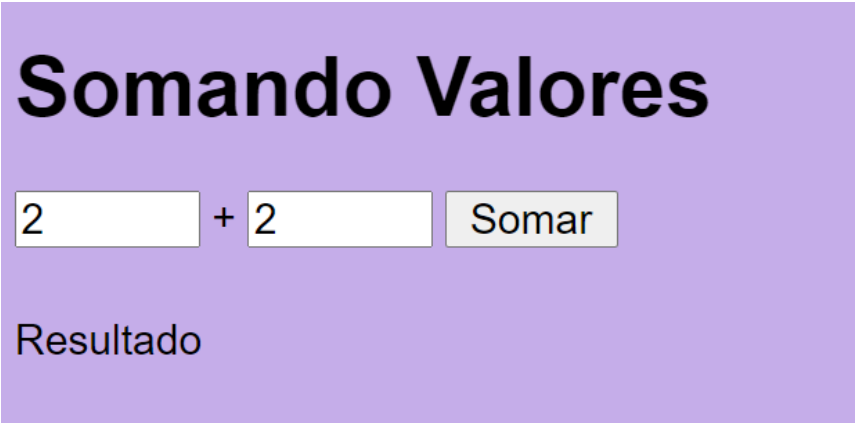
Obs.: a ferramenta "Inspecionar" / DevTools do navegador aponta erros que o JS não tem costume de apontar.

## Exercício 07 da aula 10 (Soma de valores)

Código:

```
23 <body>
24   <h1>Somando Valores</h1>
25   <input type="number" name="txtn1" id="txtn1"> +
26   <input type="number" name="txtn2" id="txtn2">
27   <input type="button" value="Somar" onclick="somar()"> <!-- Quando o botão "Somar" for clicado, a função somar() será chamada e as
    devidas alterações serão feitas de acordo a mesma -->
28   <div id="res">Resultado</div>
29
30   <!-- Mexendo com JS -->
31   <script>
32     // Criando a função para somar valores
33     function somar(){
34       var tn1 = window.document.getElementById('txtn1') // pega o valor do elemento txtn1
35       var tn2 = window.document.getElementById('txtn2') // pega o valor do elemento txtn2
36       var res = window.document.getElementById('res') // pega o conteúdo que está dentro da div 'res'
37       /* Outra maneira de usar:
38       var tn2 = window.document.querySelector('input#txtn2') // pega o valor do elemento txtn2
39       */
40
41       // Convertendo o valor dos txts de textos para números:
42       var n1 = Number(tn1.value)
43       var n2 = Number(tn2.value)
44
45       // Somando os valores:
46       s = n1 + n2
47
48       // Apresentando a soma na div 'res':
49       res.innerHTML = 'A soma entre ${n1} e ${n2} é igual a <strong>${s}</strong>.'
50     }
51   </script>
```

Resultado:



The screenshot shows a web application with a purple background. At the top, the title "Somando Valores" is displayed in a large, bold, black font. Below the title, there is a form consisting of two input fields, each containing the number "2", followed by a plus sign "+", another input field containing "2", and a button labeled "Somar". Below this form, the word "Resultado" is written in a large, black font.

# Somando Valores

2 + 2 Somar

A soma entre 2 e 2 é igual a 4.

## Módulo D: Condições em JavaScript

### ▼ Aula 11) Condições (Parte 1)

#### Tipos de condições

##### Condições simples:

Só têm 1 condição. Estrutura:

```
if (condição){  
    // condição verdadeira  
}  
// aqui, se a condição for falsa, nada acontecerá
```

##### Condições compostas:

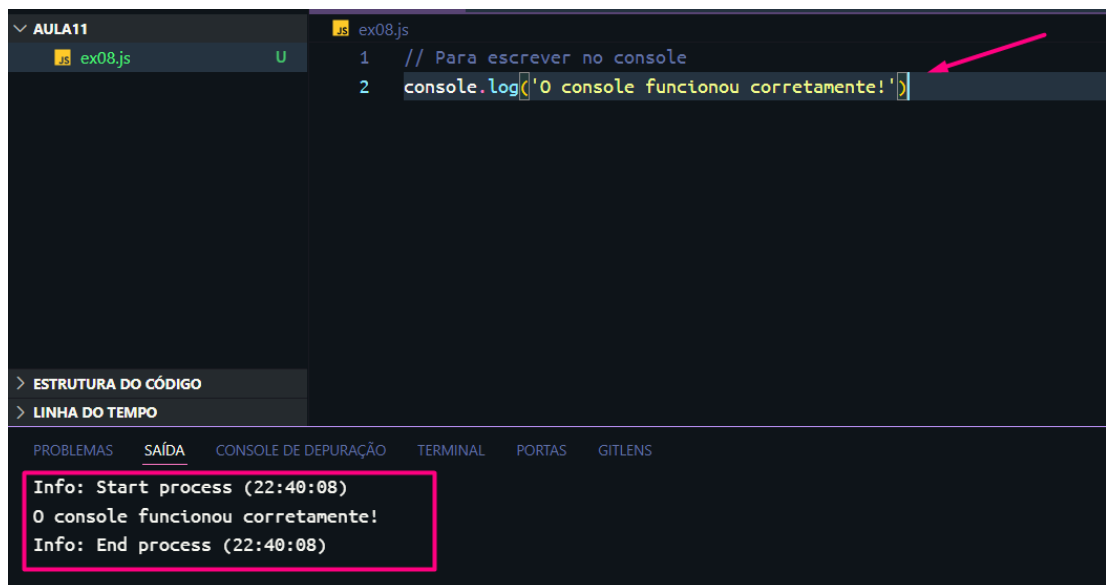
Têm + de 1 condição. Estrutura:

```
if (condição) {  
    // se a condição for verdadeira  
}  
else {  
    // se a condição for falsa  
}
```

## Mexendo com JS de forma SEQUENCIAL

Pasta de exercícios relacionados: aula11

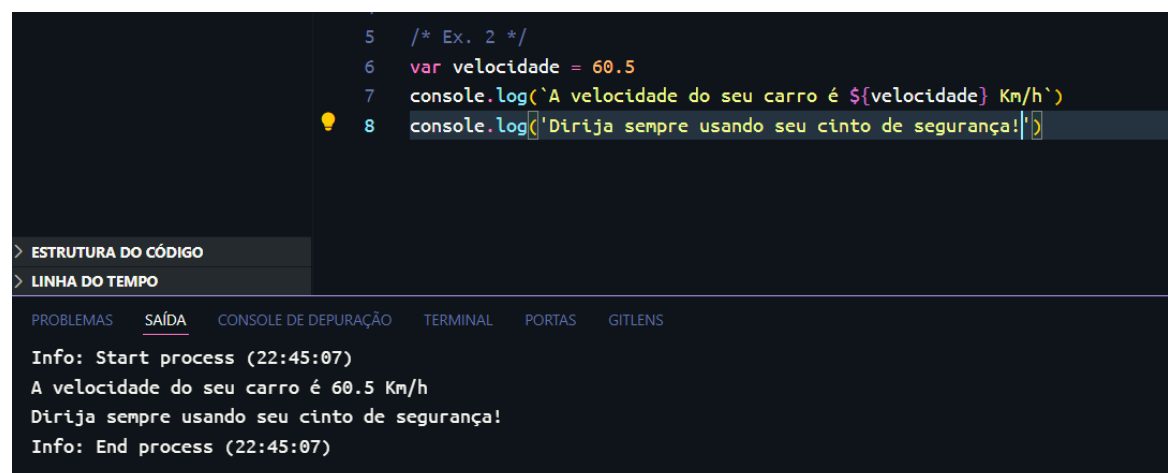
1. Crie um arquivo chamado ex08.js (sim, extensão JS!)
2. Digite o seguinte código e tecle F8, para rodar a extensão instalada no VS que roda JS no NodeJS:



```
1 // Para escrever no console
2 console.log('O console funcionou corretamente!');
```

Info: Start process (22:40:08)  
O console funcionou corretamente!  
Info: End process (22:40:08)

Ex. 2:



```
5 /* Ex. 2 */
6 var velocidade = 60.5
7 console.log(`A velocidade do seu carro é ${velocidade} Km/h`)
8 console.log('Dirija sempre usando seu cinto de segurança!');
```

Info: Start process (22:45:07)  
A velocidade do seu carro é 60.5 Km/h  
Dirija sempre usando seu cinto de segurança!  
Info: End process (22:45:07)

## Mexendo com condições no JS

Condição simples:



```
13 // Trabalhando com if-else | Condição simples
14 var velocidade = 60.5
15 console.log(`A velocidade do seu carro é ${velocidade} Km/h`)
16 if (velocidade > 60){ // Se a condição for verdadeira
17     console.log('Você está ultrapassando a velocidade permitida. MULTADO!')
18 }
19 console.log('Dirija sempre usando seu cinto de segurança!')
```

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS GITLENS

Info: Start process (22:51:08)  
A velocidade do seu carro é 60.5 Km/h  
Você está ultrapassando a velocidade permitida. MULTADO!  
Dirija sempre usando seu cinto de segurança!  
Info: End process (22:51:09)

Condição composta: ex09.js

Se a condição for verdadeira:

```
1 // Condicional composta
2 var pais = 'Brasil'
3 console.log(`Vivendo em ${pais}`)
4 if (pais == 'Brasil') {
5     console.log('Você é brasileiro(a)!')
6 }
7 else {
8     console.log('Você é estrangeiro(a)!')
9 }
```

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS GITLENS

Info: Start process (22:56:43)  
Vivendo em Brasil  
Você é brasileiro(a)!  
Info: End process (22:56:43)

Se a condição for falsa:

```
▼ AULA11
JS ex08.js U
JS ex09.js U

1 // Condicional composta
2 var pais = 'Suécia'
3 console.log(`Vivendo em ${pais}`)
4 if (pais == 'Brasil') {
5     console.log('Você é brasileiro(a)!')
6 }
7 else {
8     console.log('Você é estrangeiro(a)!')
9 }

> ESTRUTURA DO CÓDIGO
> LINHA DO TEMPO

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS GITLENS

Info: Start process (22:57:43)
Vivendo em Suécia
Você é estrangeiro(a)!
Info: End process (22:57:43)
```

### 3o exercício (HTML + JS)

Exercício: ex10.html

Código:

```
8 <body>
9   <h1>Sistema de Multas</h1>
10  Velocidade do carro: <input type="number" name="txtvel" id="txtvel"> Km/h
11  <input type="button" value="Verificar" onclick="calcular()"> <!-- Chama a função calcular() quando o botão for pressionado-->
12  <div id="res">
13    <!-- A resposta da verificação aparecerá aqui -->
14  </div>
15  <script>
16    // criando a função calcular()
17    function calcular() {
18      var txtv = window.document.querySelector('input#txtvel') // faz referência ao input cujo id é 'txtvel'
19      var res = window.document.querySelector('div#res') // faz referência a div cujo id é 'res'
20      var vel = Number(txtv.value) // txtv.value acessa a propriedade 'value' do input, converte pra número e armazena na variável vel
21
22      // alterando o que está dentro da div 'res'
23      res.innerHTML = `<p>Sua velocidade atual é de <strong>${vel}Km/h</strong></p>` // o 'p' está dentro da div 'res'
24      // pra saber se a pessoa foi multada ou não
25      if (vel > 60) { // se a condição
26        res.innerHTML += `<p>Você foi <strong>multado(a)</strong> por EXCESSO de velocidade! 🚫</p>` // o += é para não apagar a mensagem anterior
27      }
28      else { // se a condição for falsa
29        res.innerHTML += `<p>Você está dirigindo dentro do limite permitido! 🟢</p>`
30      }
31      res.innerHTML += `<p>Dirija sempre com cinto de segurança!</p>`
32    }
33  </script>
34 </body>
```

Execução se a condição for verdadeira:

# Sistema de Multas

Velocidade do carro:  Km/h Verificar

Sua velocidade atual é de **68Km/h**

Você foi **multado(a)** por EXCESSO de velocidade! 🚫

Dirija sempre com cinto de segurança!

Execução se a condição for falsa:

# Sistema de Multas

Velocidade do carro:  Km/h Verificar

Sua velocidade atual é de **55Km/h**

Você está dirigindo dentro do limite permitido! 👍

Dirija sempre com cinto de segurança!

## ▼ Aula 12) Condições (Parte 2)

Lembrando que:

- Condição simples:

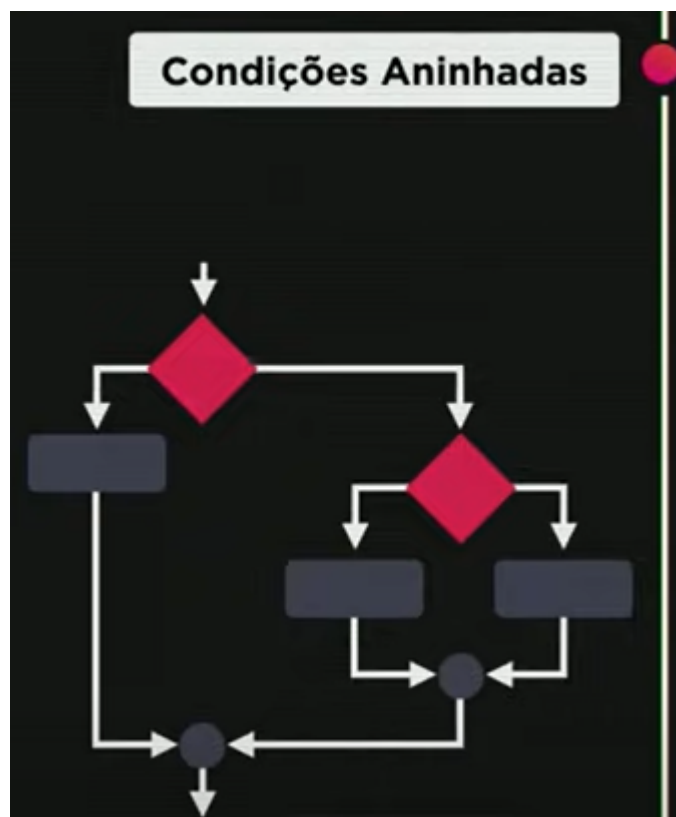
```
if (condição){  
    // true  
}
```

- Condição composta:

```
if (condição){  
    // true  
}  
else {  
    // false  
}
```

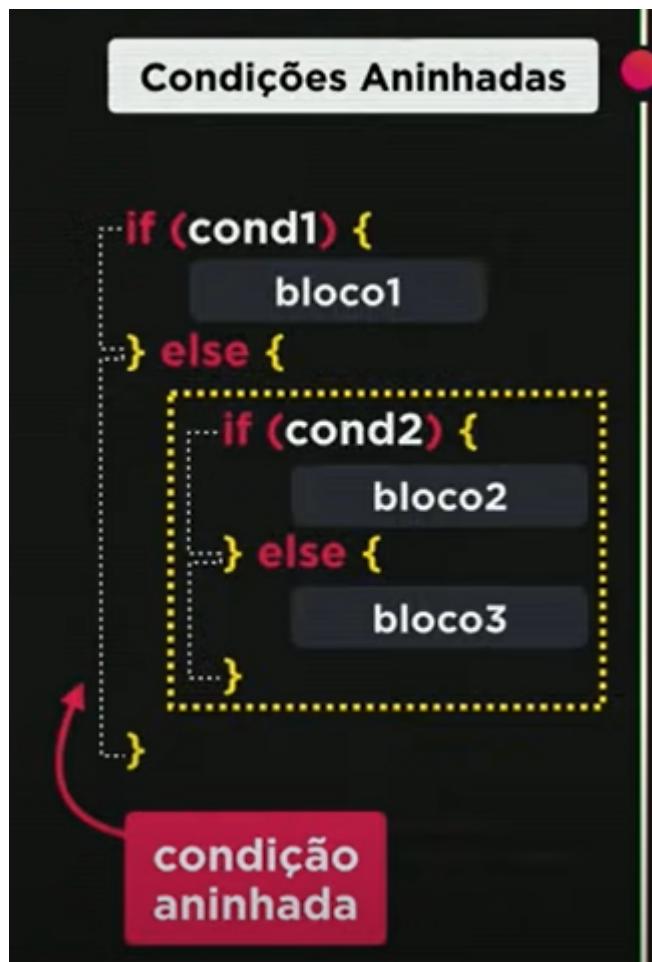
## Condição aninhada

Condições aninhadas: condição DENTRO de condição. Exemplo:



Fluxograma representando as condições aninhadas. Fonte: Curso em Vídeo.

Estrutura de condições aninhadas em código:



Estrutura de código de condições aninhadas. Fonte: Curso em Vídeo.

Numa condição aninhada, a condição 2 só é executada se a primeira condição for falsa.

Usar `console.log` quando for usar o `NodeJS`.

Bloco encadeado no JS:

```
if (condição 1){  
    // critério 1  
}  
else if (condição 2){  
    // critério 2  
}  
else {  
    // critério 3  
}
```

Ex. 1 de condicional encadeada em JS:

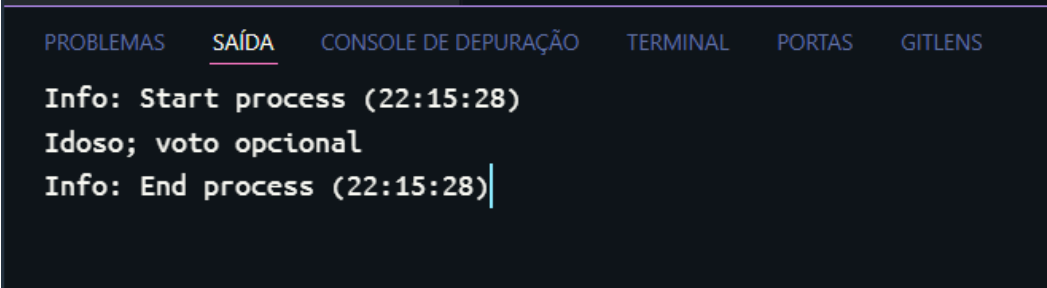
Código:

```
// Dizendo se a pessoa pode votar ou não dependendo da sua idade

var idade = 65

// Condicionais encadeadas
if (idade < 16) {
    console.log('Criança/adolescente; não vota')
}
else if (idade == 16 || idade == 17){
    console.log('Jovem; voto opcional')
}
else if (idade >= 18 && idade <= 64){
    console.log('Adulto; voto obrigatório')
}
else {
    console.log('Idoso; voto opcional')
}
```

Resultado da execução:



```
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  PORTAS  GITLENS

Info: Start process (22:15:28)
Idoso; voto opcional
Info: End process (22:15:28)|
```

Ex. 2 de condicional encadeada em JS:

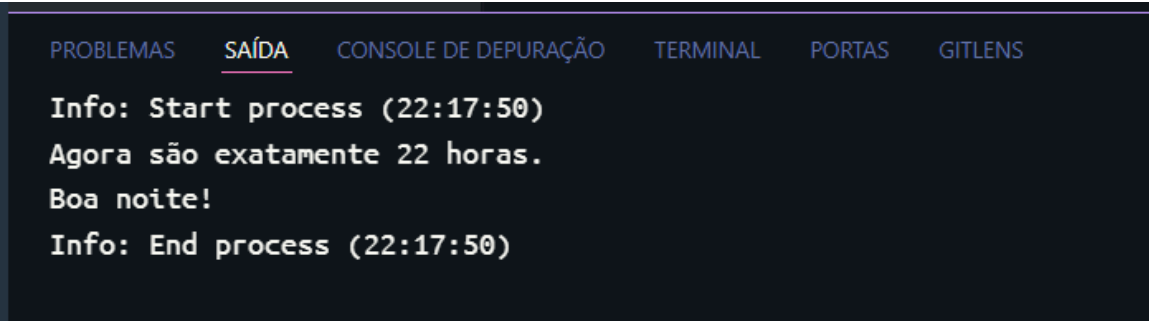
Código:

```
// Dando bom dia, boa tarde, boa noite ou boa madrugada
dependendo da hora informada

// Pra pegar a hora atual do sistema
var agora = new Date()
var hora = agora.getHours() // pega a hora atual
console.log(`Agora são exatamente ${hora} horas.`)

// Verificando o período da hora
if (hora < 12){
    console.log('Bom dia!')
}
else if (hora <= 18){
    console.log('Boa tarde!')
}
else if (hora <= 23){
    console.log('Boa noite!')
}
else [
    console.log('Boa madrugada! Zzz')
]
```

Resultado da execução:



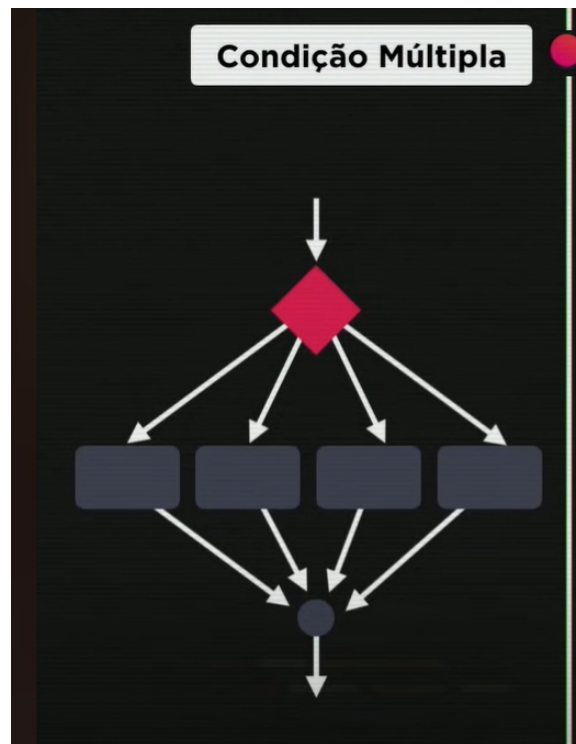
```
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  PORTAS  GITLENS

Info: Start process (22:17:50)
Agora são exatamente 22 horas.
Boa noite!
Info: End process (22:17:50)
```

## Condição múltipla

Tem a possibilidade de não ser só sim ou não, mas também de outros valores. Não é aplicada em muitas situações, somente para específicas. É

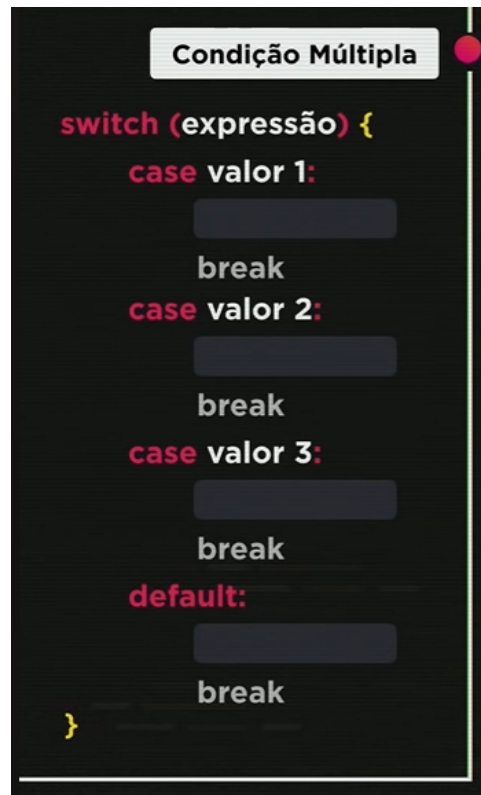
importante para valores pontuais; é mais limitado. Só funciona com CARACTERES e STRINGS.



Representação da condição múltipla. Fonte: Curso em Vídeo.

Estrutura do código de condição múltipla:





Estrutura de código da condição múltipla. Fonte: Curso em Vídeo.

Toda condição `switch case` precisa ter um `break` ao final!

Ex. de código de condição múltipla:

```
// Programa que mostra o dia atual da semana  
  
var agora = new Date()  
var diaSem = agora.getDay() // pega o dia atual da semana  
a  
  
/*  
Lembrando que, no JS:  
0- Domingo  
1- Segunda  
2- Terça  
3- Quarta  
4- Quinta  
5- Sexta  
6- Sábado
```

```

*/

console.log(diaSem) // Mostra o dia da semana

// Condições múltiplas (específicas)
switch (diaSem){
  case 0:
    console.log('Domingo')
    break
  case 1:
    console.log('Segunda-feira')
    break
  case 2:
    console.log('Terça-feira')
    break
  case 3:
    console.log('Quarta-feira')
    break
  case 4:
    console.log('Quinta-feira')
    break
  case 5:
    console.log('Sexta-feira')
    break
  case 6:
    console.log('Sábado')
    break
  default:
    console.log('[ERRO] Dia inválido!') // para valo
res inválidos
    break
}

```

Resultado da execução:

```
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  PORTAS  GITLENS

Info: Start process (09:13:32)
2
Terça-feira
Info: End process (09:13:32)
```

Exercícios relacionados a aula: ex011.js, ex012.js e ex013.js

## ▼ Exercícios Aula 12 (parte 1, parte 2 e parte 3)

Exercício ex014:

Código HTML:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Modelo de Exercício</title>
  <link rel="stylesheet" href="style.css">
</head>
<body onload="carregar()"> <!-- Quando a página carregar, chamar a função carregar() -->
  <!-- Cabeçalho -->
  <header>
    <h1>Hora do dia</h1>
  </header>
  <!-- Seção -->
  <section>
    <!-- div 1 -->
    <div id="msg">
      Aqui vai aparecer a mensagem
    </div>
    <!-- div 2 -->
    <div id="foto">
      
        </div>
    </section>
    <!-- Rodapé -->
    <footer>
        <p>&copy; GabriellaXavier</p> <!-- Símbolo de co
pyright-->
    </footer>
    <script src="script.js"></script> <!-- CSS externo -
->
</body>
</html>

```

Código JS:

```

function carregar(){
    var msg = window.document.getElementById('msg') // r
eferência com o id 'msg' da div 1
    var imagem = window.document.getElementById('image
m') // referência com o id 'foto' da div 2

    // para pegar ao horário atual:
    var data = new Date()
    var hora = data.getHours()
    var min = data.getMinutes()

    // para apresentar a mensagem ao usuário
    msg.innerHTML = `Agora são ${hora}h e ${min}min.`

    // criando as condições
    if (hora >= 0 && hora < 12){
        // Bom dia; entre 00h e 11h59
        imagem.src = 'imgs/manha.png'
        document.body.style.background = '#fce38d' // mu
dando o fundo da página de acordo com a foto exibida
    }
    else if (hora >= 12 && hora <= 18){
        // Boa tarde; entre 12h e 18h
    }
}

```

```

        imagem.src = 'imgs/tarde.png'
        document.body.style.background = '#e88552' // mu
dando o fundo da página de acordo com a foto exibida
    }
    else {
        // Boa noite; acima das 18h
        imagem.src = 'imgs/noite.png'
        document.body.style.background = '#708685' // mu
dando o fundo da página de acordo com a foto exibida
    }
}

```

Resultado da execução no navegador:



ex015:

Código HTML:

```

<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, i
nitial-scale=1.0">
    <title>Verificador de Idade</title>
    <link rel="stylesheet" href="style.css">

```

```

</head>
<body>
  <!-- Cabeçalho -->
  <header>
    <h1>Título</h1>
  </header>
  <!-- Seção -->
  <section>
    <!-- div 1 -->
    <div>
      <p>
        Ano de nascimento:
        <input type="number" name="txtano" id="txtano" min="0">
      </p>
      <p>
        Sexo:
        <input type="radio" name="radsex" id="masc" checked>
        <label for="masc">Masculino</label>
        <input type="radio" name="radsex" id="femi">
        <label for="femi">Feminino</label>
      </p>
      <p>
        <input type="button" value="Verificar" onclick="verificar()"> <!-- quando o botão for pressionado, a função verificar() será chamada -->
      </p>
    </div>
    <!-- div 2 -->
    <div id="res">
      Preencha os dados acima para ver o resultado!
    </div>
  </section>
  <!-- Rodapé -->
  <footer>

```

```

        <p>&copy; GabriellaXavier</p> <!-- Símbolo de co
pyright-->
    </footer>
    <script src="script.js"></script> <!-- JS externo --
>
</body>
</html>

```

Código JS:

```

// Criando a função para verificar a idade do usuário
function verificar(){
    var data = new Date()
    var ano = data.getFullYear() // Vai pegar o ano no f
ormato yyyy
    var fano = document.getElementById('txtano') // Refe
rência ao campo txtano, onde o usuário informa o ano de
nascimento
    var res = document.querySelector('div#res') // Refer
ência ao id 'res'

    // Verifiicando se o ano informado é maior que o ano
atual
    if (fano.value.length == 0 || Number(fano.value) > a
no){ // Se o ano informado for igual a 0 ou for maior qu
e o ano atual...
        window.alert('[ERRO] Verifique os dados e tente
novamente!')
    }
    else {
        var fsex = document.getElementsByName('radsex')
// Referência ao radiobutton
        var idade = ano - Number(fano.value) // Descobri
ndo a idade do usuário
        var genero = ''

        // Inserindo imagem na página HTML usando o JS
        var img = document.createElement('img')

```

```

        img.setAttribute('id', 'foto') // Cria um id par
a a tag img

        // Verificando o gênero da pessoa
        if (fsex[0].checked){ // Se a opção 0 do radiobu
tton foi selecionada, então...
            genero = 'Homem'
            // Apresentando uma imagem de acordo com a f
aixa etária do usuário
            if (idade >= 0 && idade <=10){
                // Criança
                img.setAttribute('src', 'imgs/bb_menino.
png') // apresenta a img do bebê menino
            }
            else if (idade < 21){
                // Jovem
                img.setAttribute('src', 'imgs/rapaz.pn
g') // apresenta a img do rapaz
            }
            else if (idade < 60){
                // Adulto
                img.setAttribute('src', 'imgs/homem_adul
to.png') // apresenta a img do homem adulto
            }
            else {
                // Idoso
                img.setAttribute('src', 'imgs/idoso.pn
g') // apresenta a img do idoso
            }
        }
        else if (fsex[1].checked){ // Se a opção 1 do ra
diobutton foi selecionada, então...
            genero = 'Mulher'
            // Apresentando uma imagem de acordo com a f
aixa etária do usuário
            if (idade >= 0 && idade <=10){
                // Criança
                img.setAttribute('src', 'imgs/bb_menina.

```



```

    png') // apresenta a img da bebê menina
    }
    else if (idade < 21){
        // Jovem
        img.setAttribute('src', 'imgs/moca.png')
    // apresenta a img da moça
    }
    else if (idade < 60){
        // Adulto
        img.setAttribute('src', 'imgs/mulher-adulta.png') // Apresenta a img da mulher adulta
    }
    else {
        // Idoso
        img.setAttribute('src', 'imgs/idosa.png') // apresenta a imagem do idosa
    }
    }
    res.innerHTML = `Detectamos ${genero} com ${idade} anos.`
    res.appendChild(img) // Adicionando o elemento do tipo img na div 'res'
  }
}

```


Resultado da execução:

### Verificador de Idade

Ano de nascimento:

Sexo: ☒ Masculino ☐ Feminino

Detectamos Homem com 1 anos.

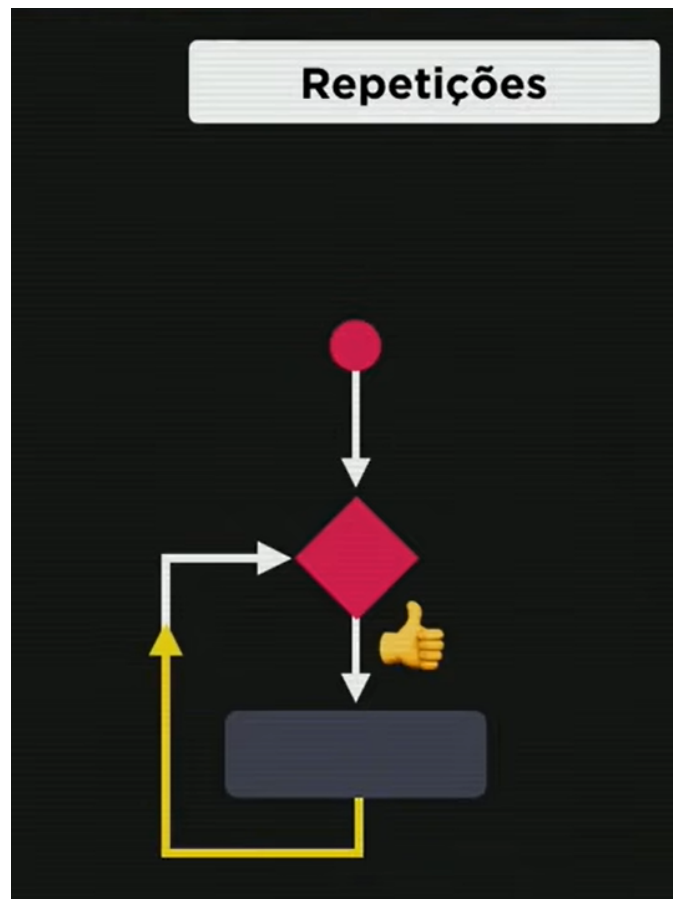


© GabriellaXavier

## Módulo E: Repetições em JavaScript

### ▼ Aula 13 - Repetições (Parte 1)

O laço de repetição é outro tipo de estrutura de controle na programação. Testa condições também, mas, diferente do if-else, repete a execução de um bloco.



Estrutura de um laço de repetição. Fonte: Curso em Vídeo.

Enquanto a condição for verdadeira, em um loop, o bloco de código continuará a ser executado.

Usar laços de repetição reduzem o número de linhas de código. 😊

## Estruturas de repetição

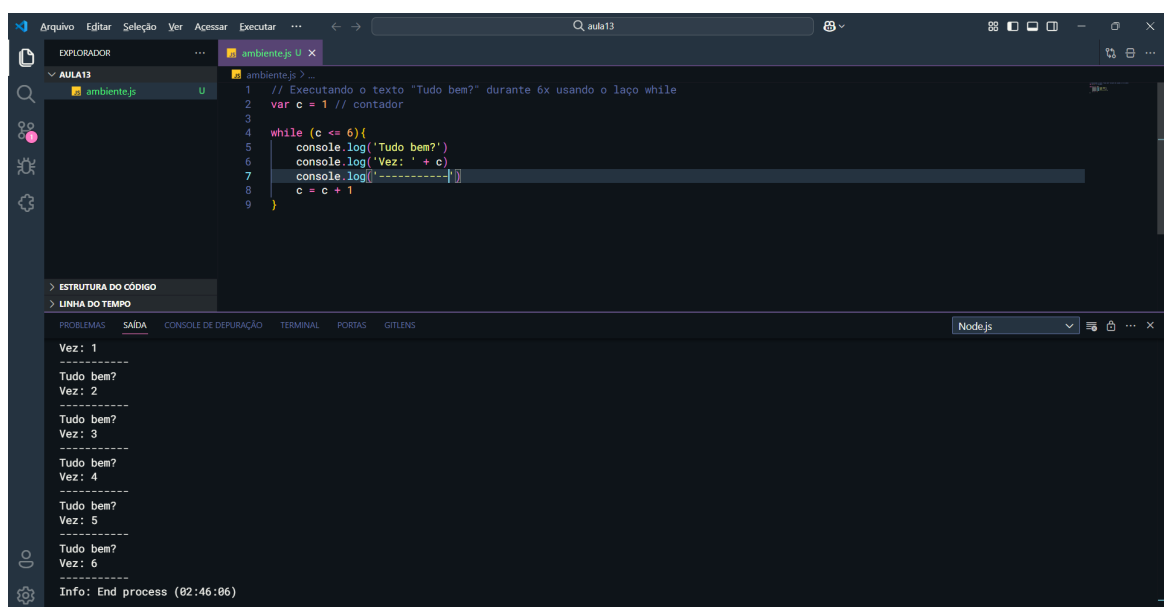
```
while (condição) {  
    // código  
} // se a condição for verdadeira, o laço continua a se  
repetir. Se for falsa, para.
```

```
// Estrutura do while  
do {  
    // código  
} while (condição) // no do while, a condição explicada  
ao final do bloco do código
```

```
// Estrutura for  
for (contador; condição; incremento){  
    // código  
} // no for, tudo é declaração na mesma linha
```

while = ENQUANTO

1o exercício e execução (usando `while`):



The screenshot shows a code editor with a file named 'aula13' open. The code defines a variable 'c' and uses a 'while' loop to print 'Tudo bem?' six times, incrementing 'c' each time. The console output shows the execution of the loop, printing the message and the current value of 'c' from 1 to 6. The process ends with the message 'Info: End process (02:46:06)'.

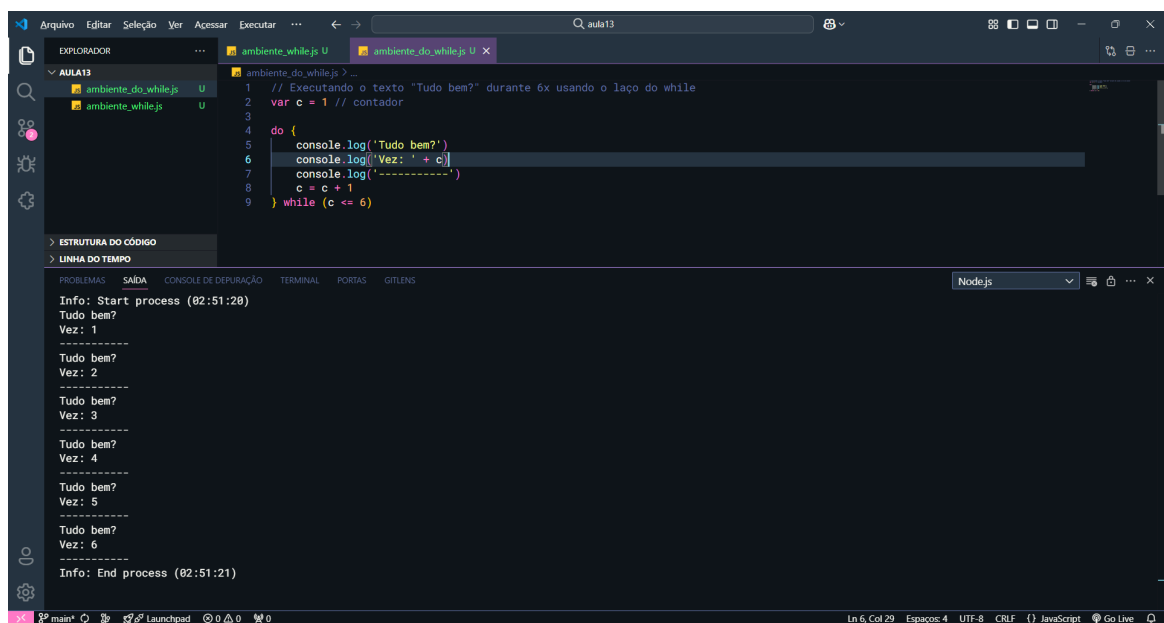
```
1 // Executando o texto "Tudo bem?" durante 6x usando o laço while  
2 var c = 1 // contador  
3  
4 while (c <= 6){  
5     console.log('Tudo bem?')  
6     console.log('Vez: ' + c)  
7     console.log('-----')  
8     c = c + 1  
9 }
```

Veja a saída no console:

```
Ve: 1  
Tudo bem?  
Ve: 2  
Tudo bem?  
Ve: 3  
Tudo bem?  
Ve: 4  
Tudo bem?  
Ve: 5  
Tudo bem?  
Ve: 6  
Info: End process (02:46:06)
```

Fonte: Compilação da autora.

2o exercício e execução (usando `do while`):



```
1 // Executando o texto "Tudo bem?" durante 6x usando o laço do while
2 var c = 1 // contador
3
4 do {
5     console.log("Tudo bem?")
6     console.log("Vez: " + c)
7     console.log("-----")
8     c = c + 1
9 } while (c <= 6)
```

Info: Start process (02:51:20)  
Tudo bem?  
Vez: 1  
-----  
Tudo bem?  
Vez: 2  
-----  
Tudo bem?  
Vez: 3  
-----  
Tudo bem?  
Vez: 4  
-----  
Tudo bem?  
Vez: 5  
-----  
Tudo bem?  
Vez: 6  
-----  
Info: End process (02:51:21)

## Outras informações sobre JS

### ▼ `const`

É um tipo de variável que, uma vez atribuído um valor a ela, não dá para alterar. É como se fosse fixo.

### ▼ `let`

- **Escopo de Bloco:** A variável existe apenas no bloco onde foi declarada (entre `{ }`).

```
if (true) {
    let nome = "João";
    console.log(nome); // "João"
}
console.log(nome); // Erro: nome is not defined
```

- **Reatribuição permitida:** Você pode atribuir novos valores à variável.

```
let idade = 25;
idade = 30;
console.log(idade); // 30
```

- **Não permite declarações duplicadas** no mesmo escopo.

```
let cidade = "São Paulo";
let cidade = "Rio de Janeiro"; // Erro: Identifier
'cidade' has already been declared
```

- **Temporal Dead Zone (TDZ)**: A variável não pode ser acessada antes de ser declarada no código.

```
console.log(x); // Erro: Cannot access 'x' before ini
tialization
let x = 10;
```

## ▼ Diferenças `var` x `let`

A principal diferença entre `let` e `var` em JavaScript é o **escopo** e o **comportamento** de declaração.

### 1. Escopo:

- `let` tem **escopo de bloco**, ou seja, a variável declarada com `let` é acessível apenas dentro do bloco `{ }` onde foi definida (como em um `if`, `for` ou função).
- `var` tem **escopo de função**, o que significa que a variável declarada com `var` é acessível em toda a função, mesmo que declarada dentro de um bloco (como um `if`).

### 2. Declaração duplicada:

- `let` não permite que a mesma variável seja declarada mais de uma vez no mesmo escopo.

- `var` permite declarar a mesma variável várias vezes no mesmo escopo, o que pode causar confusão ou erros.

### 3. Hoisting:

- `let` é "hoisted" (elevação) para o topo do seu bloco, mas não pode ser acessado até a linha onde foi declarado, resultando na "**zona morta temporal**".
- `var` é "hoisted" para o topo da função ou escopo global e pode ser acessado antes da declaração, mas com valor `undefined`.

### Resumo:

- `let` é mais seguro e previsível, com escopo de bloco e restrição a reatribuições no mesmo escopo.
- `var` é mais antigo, tem escopo de função e permite declarações duplicadas no mesmo escopo.