

## 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
  mazenar 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)`

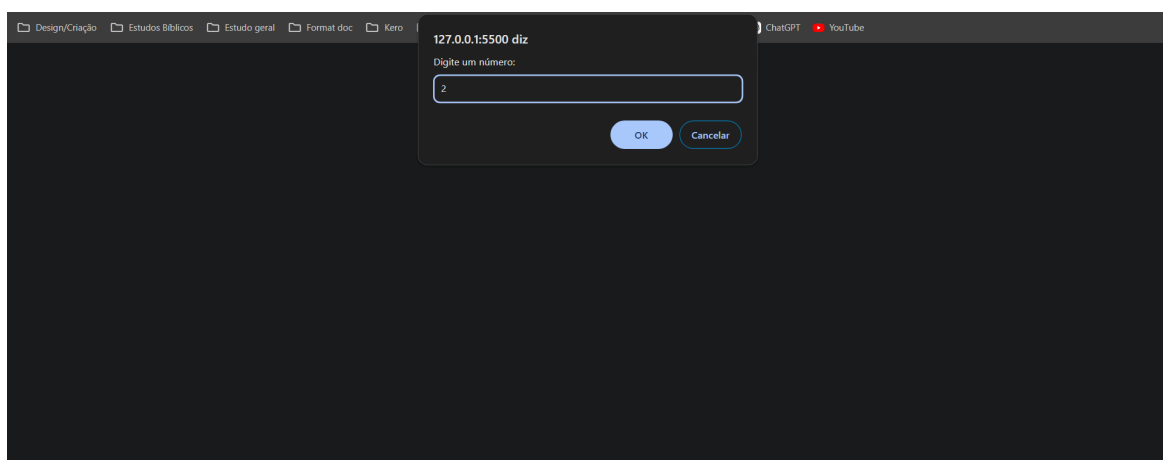
Ex. de código:

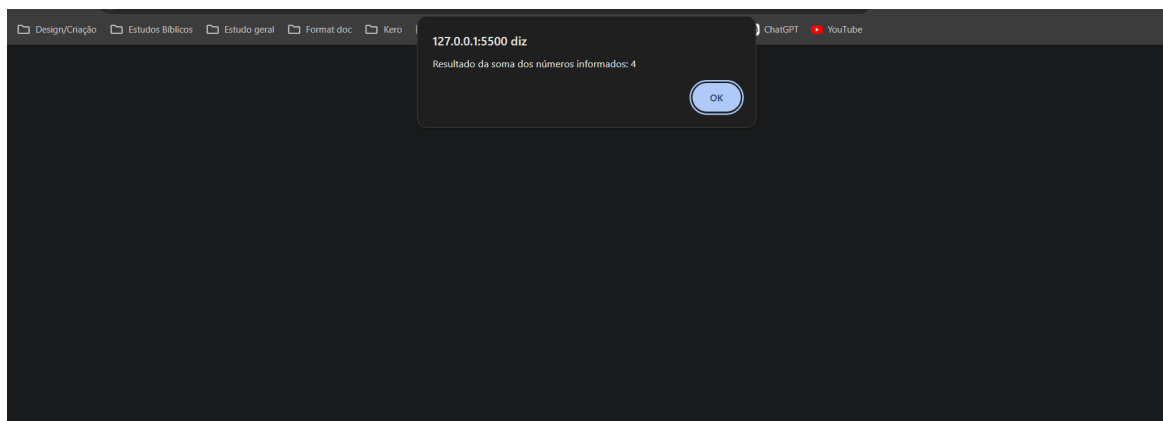
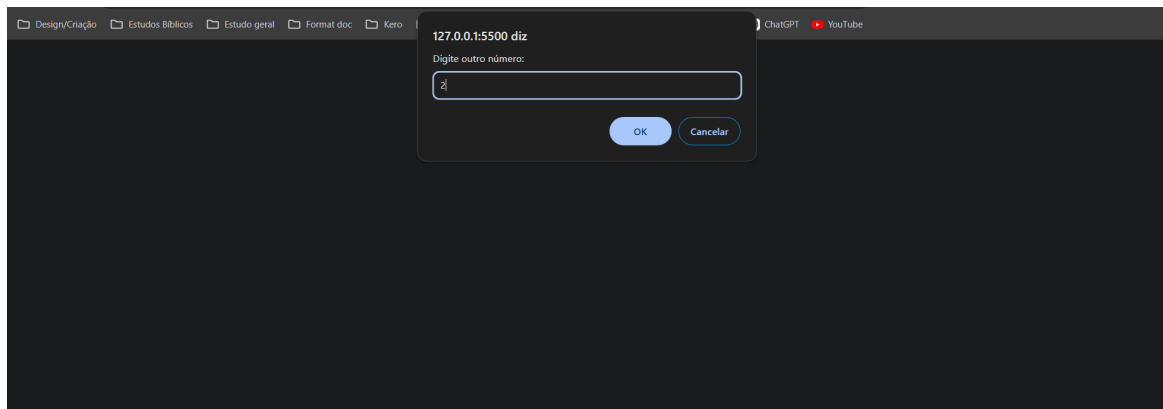
```
<script>
    // lendo 2 valores informados pelo usuário e convertendo-os em números inteiros para que a exibição da soma ocorra sem problemas
    var n1 = Number.parseInt(window.prompt('Digite um número: '))
    var n2 = Number.parseInt(window.prompt('Digite o outro número: '))

    // 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.:

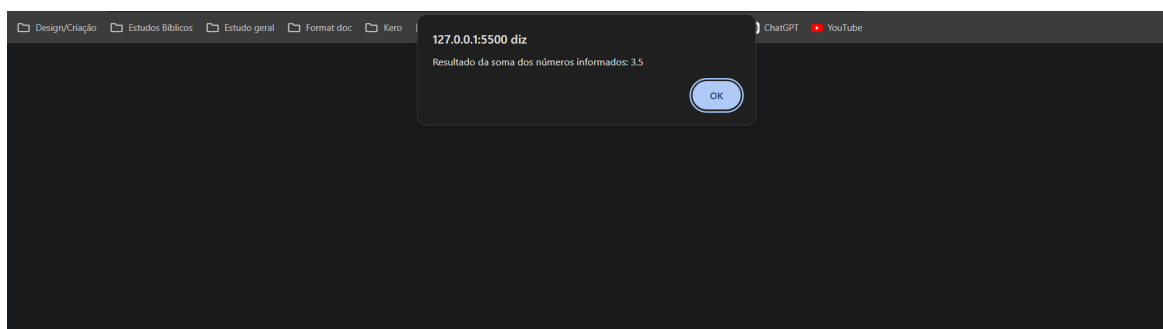
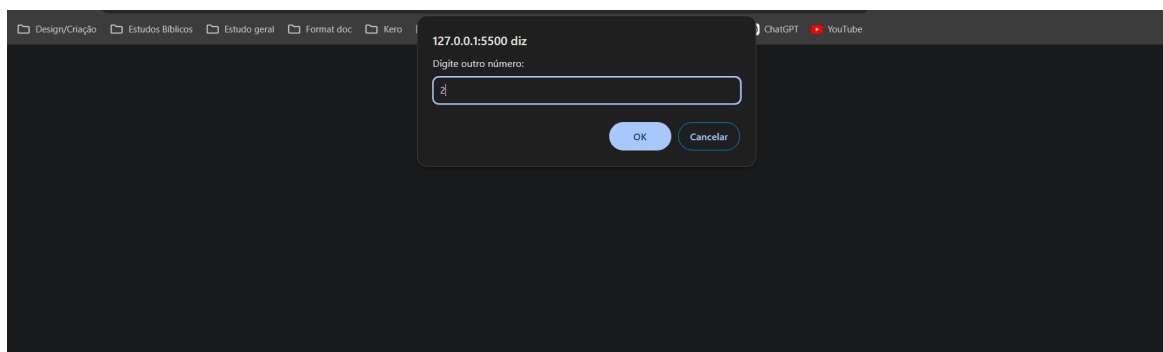
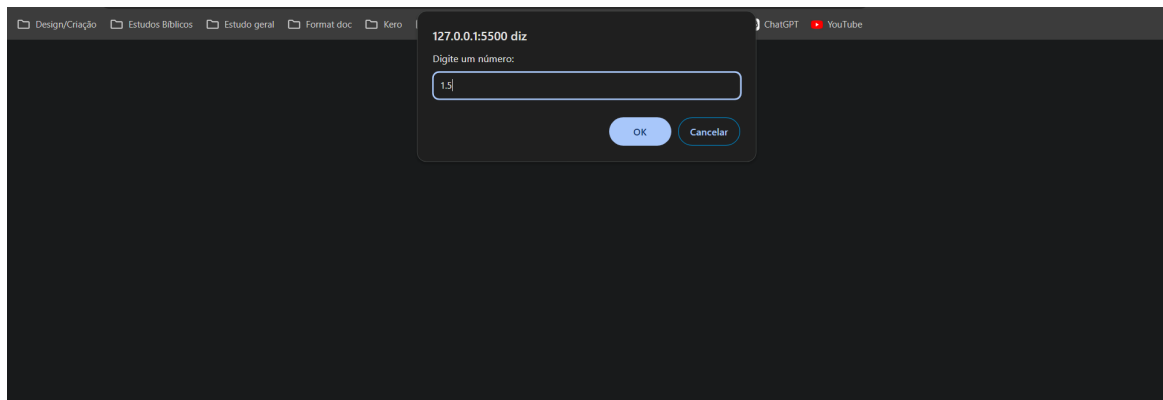
Código:

```
// 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\Documents\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:

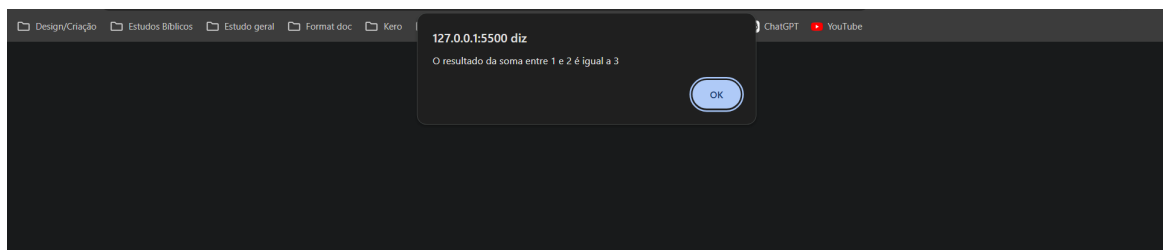
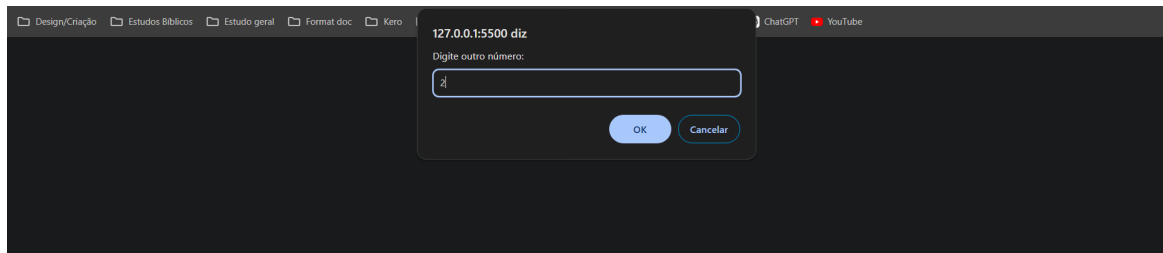
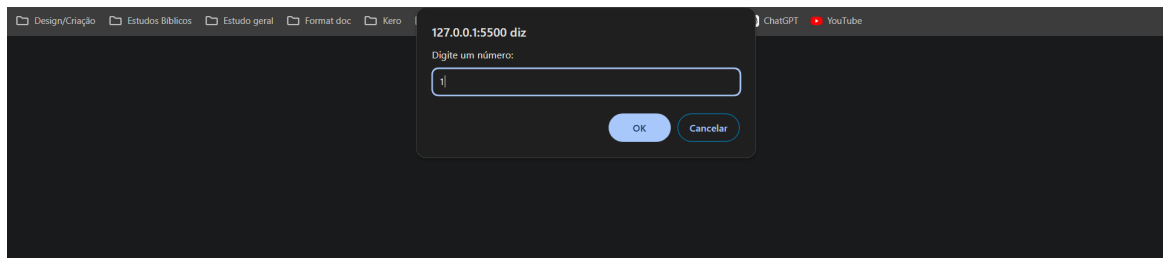
```
// Usando somente Number() para deixar o JS decidir o tipo de dado:
var n1 = Number(window.prompt('Digite um número: '))
var n2 = Number(window.prompt('Digite outro número: '))

// 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:

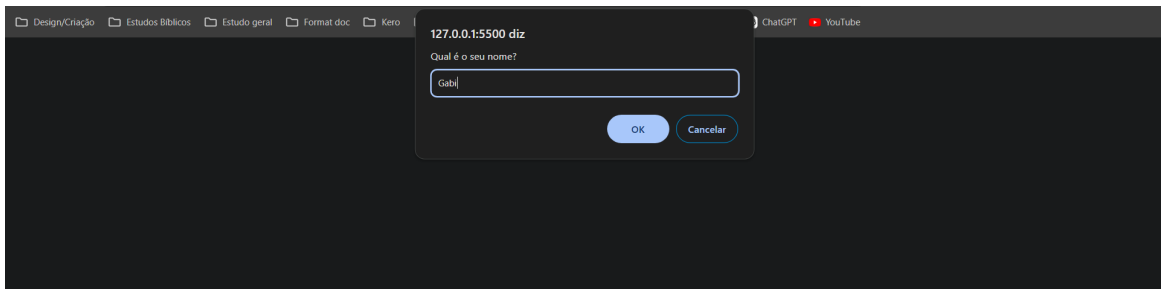
```
var s = 'JavaScript'
s.length() // quantos caracteres a String tem
s.toUpperCase() // tudo para MAIÚSCULO
s.toLowerCase() // tudo para MINÚSCULO
```

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

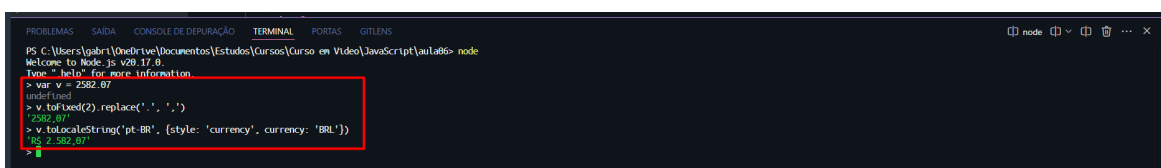
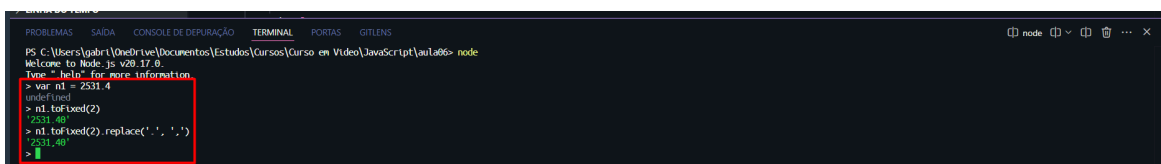
Seu nome tem 4 letras.

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:

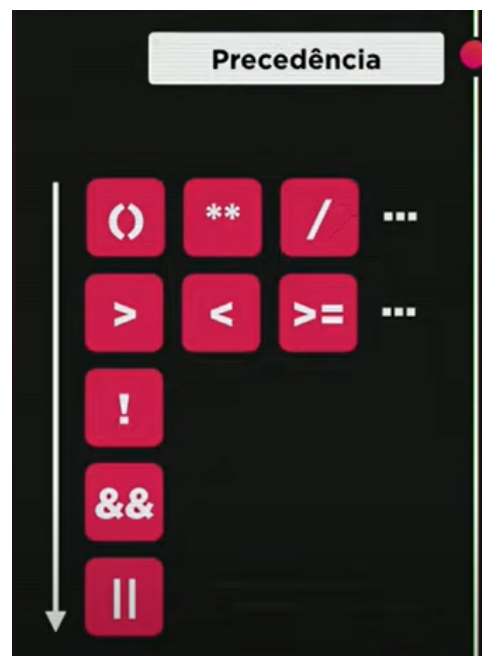
1. `!`
2. `&&`
3. `||`

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.

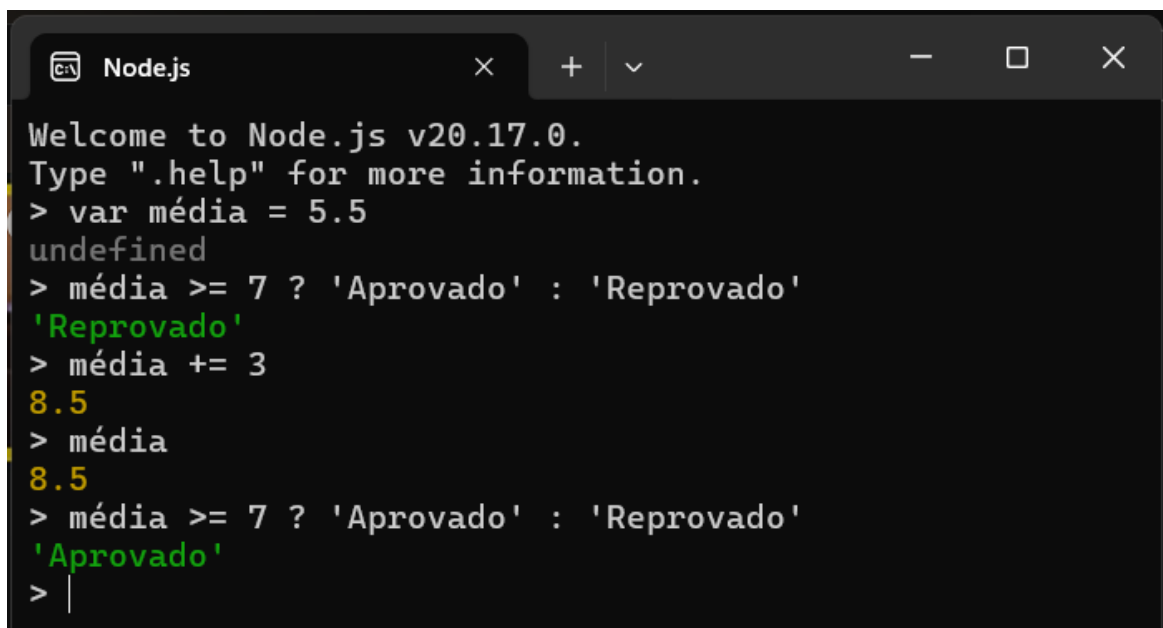
Ex.:

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

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

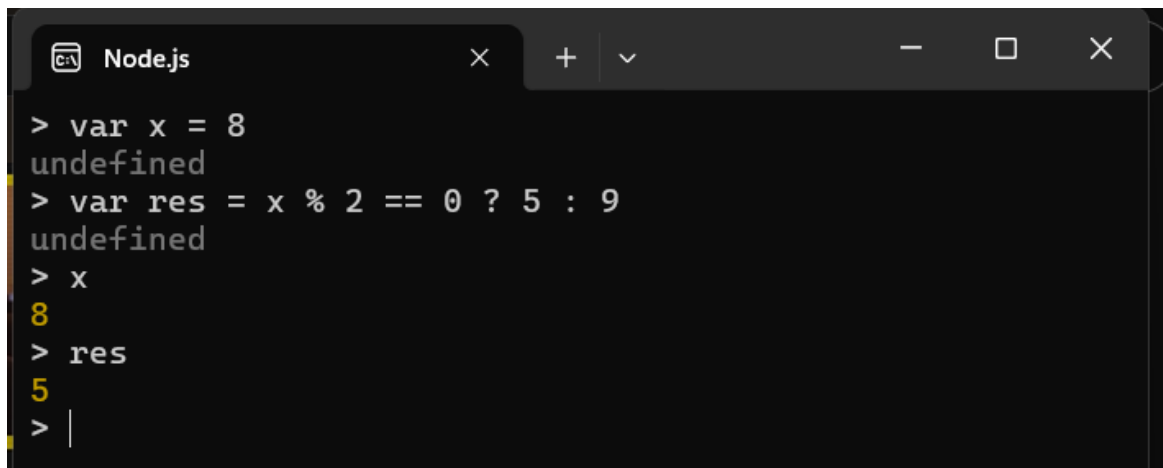
No NodeJS:

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