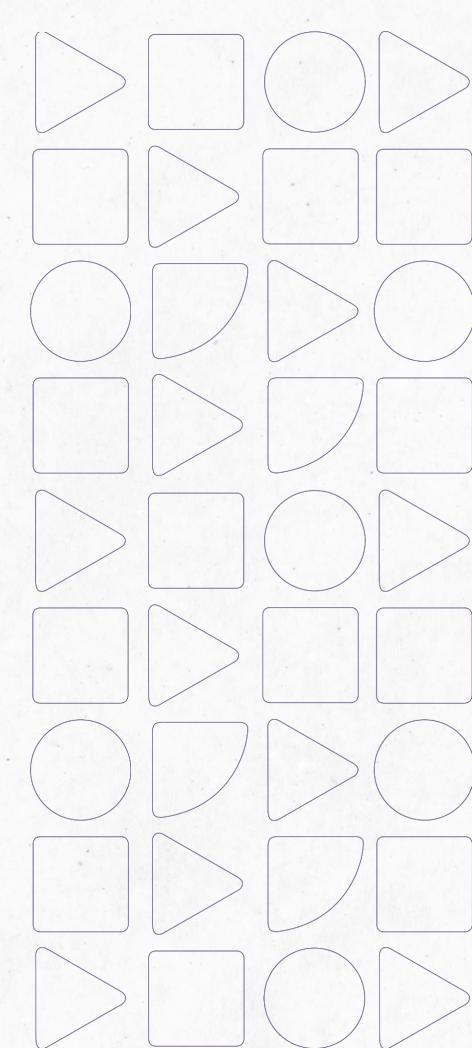


# Operadores

**Disciplina:** Linguagem de Programação



## Conteúdos:

Operadores.

## Habilidade(s):

- Compreender a lógica da matemática na linguagem de programação;
- Conhecer os símbolos importantes na sintaxe e suas funções.

# Bloco 1

---

Compreendendo o que são operadores lógicos na linguagem de Javascript.

# Como você se sente?

Quem é você na hora de resolver uma conta matemática?



1



2



3



4



5



6

# Nem todo mundo gosta de fazer conta

A matemática é uma disciplina que pode ser desafiadora e intimidante para algumas pessoas.

Na hora de fazer algum cálculo matemático, existem diferentes tipos de pessoas:



O que faz de cabeça.



O que usa calculadora.



O que conta nos dedos.



É assim que me sinto quando me dão o troco em moedas e eu preciso contar.

Mas os seus problemas acabaram!

Na linguagem Javascript, os operadores servem como símbolos ou palavras-chave que você pode usar para executar operações em valores.

Mas eles não permitem apenas realizar cálculos. Com eles, você também pode comparar valores e manipular *strings* (textos).



# Conhecendo os operadores

Imagine que JavaScript é uma linguagem de programação que permite que você realize ações com informações, como números ou palavras.

Operadores são como ferramentas que você usa para fazer determinadas atividades com essas informações.

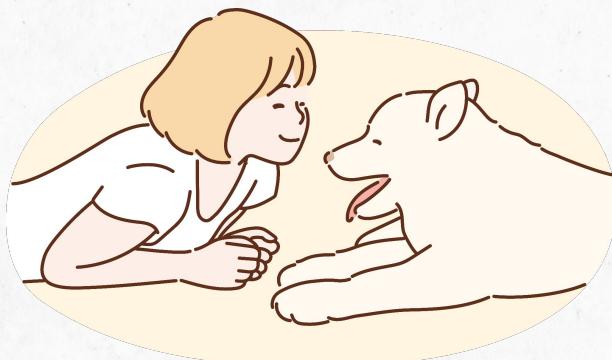
Eles são um pouco como os operadores em matemática, mas, em vez de apenas fazerem cálculos, podem ter muitas outras funções.



# Operadores de atribuição

São utilizados para dar um nome a um pedaço de informação, como um número, um texto ou qualquer outro elemento.

Isso nos ajuda a lembrar e usar essa informação mais tarde no nosso programa. Para isso, você utiliza o sinal de “=” (igual).

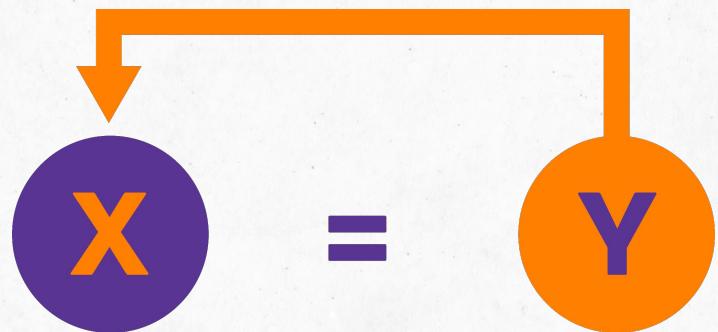


Imagine que você acabou de ganhar um cachorro e quer chamá-lo de Pelúcia.

Assim, você utiliza a função `meuCachorro = pelucia`

## Atribuindo valores

Como você pode perceber, o elemento da direita atribui valor ao elemento da esquerda. Sendo assim, a função correta para o operador de atribuição é:



# Outros tipos de operadores

Veja os principais tipos de operadores na tabela abaixo.

Nome	Operador encurtado	Significado
Atribuição	$x = y$	$x = y$
Atribuição de adição	$x += y$	$x = x + y$
Atribuição de subtração	$x -= y$	$x = x - y$
Atribuição de multiplicação	$x *= y$	$x=x*y$
Atribuição de divisão	$x /= y$	$x=x/ y$
Atribuição de resto	$x %= y$	$x = x \% y$
Atribuição exponencial	$x **= y$	$x = x ** y$

# Operadores de comparação

Os operadores de comparação são usados em programação para comparar dois valores e verificar se uma condição é verdadeira ou falsa.

Eles geralmente são usados em estruturas de controle, como condicionais (*if, else*) e *loops*, para tomar decisões com base nas comparações.



# Tipos de operadores comuns

`==` (igual a)

Esse operador verifica se dois valores são iguais.

Ex.: `5 == 5` # Isso é verdadeiro, porque 5 é **igual** a 5.

`!=` (diferente de)

Esse operador verifica se dois valores são diferentes.

Ex.: `5 != 3` # Isso é verdadeiro, porque 5 é **diferente** de 3.

`>` (maior do que)

Esse operador verifica se o valor da esquerda é maior do que o valor da direita.

Ex.: `6 > 4` # Isso é verdadeiro, porque 6 é **maior do que** 4.

`<` (menor do que)

Esse operador verifica se o valor da esquerda é menor do que o valor da direita.

Ex.: `2 < 7` # Isso é verdadeiro, porque 2 é **menor do que** 7.

`>=` (maior ou igual a)

Esse operador verifica se o valor da esquerda é maior ou igual ao valor da direita.

Ex.: `B5 >= 5` # Isso é verdadeiro, porque 5 é **maior ou igual a** 5.

`<=` (menor ou igual a)

Esse operador verifica se o valor da esquerda é menor ou igual ao valor da direita.

Ex.: `3 <= 3` # Isso é verdadeiro, porque 3 é **menor ou igual a** 3.

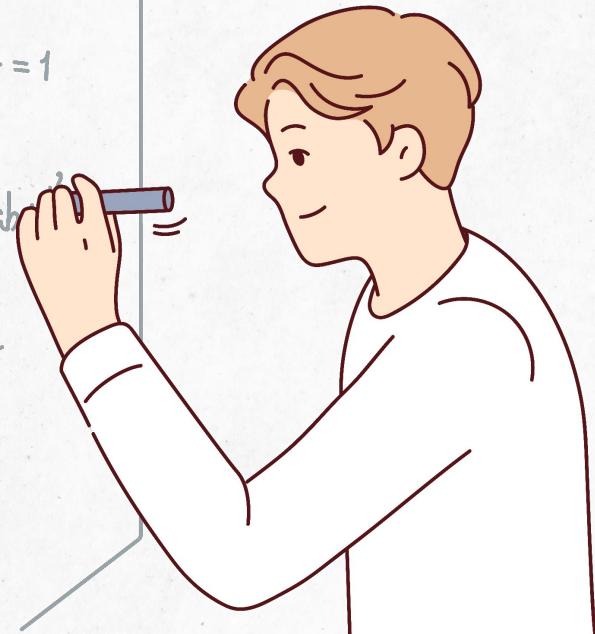
# Operadores aritméticos

São símbolos ou sinais matemáticos que você pode usar em programação para realizar cálculos matemáticos em números.

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

$$(a+b)^2 = a^2 + 2ab + b^2$$

$$x = \sqrt{\frac{b^2}{c}} + c - \frac{b}{2}$$



# Tipos comuns de operadores aritméticos

Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Exponencial	**
Resto da divisão	%
Incremento	++
Decremento	--

# Bloco 2

---

Vamos praticar o que vimos até aqui?



## Dê um *play* no conhecimento!



Vamos aprender como elaborar um mapa mental?

## Mão na massa

Separem-se em cinco grupos e analisem os casos que o professor entregar na folha de ofício.

Discutam entre si, desenvolvam um operador para a variável e criem um **mapa mental** na cartolina. Vocês podem usar cores, setas e textos para organizar as informações de forma clara e visual.



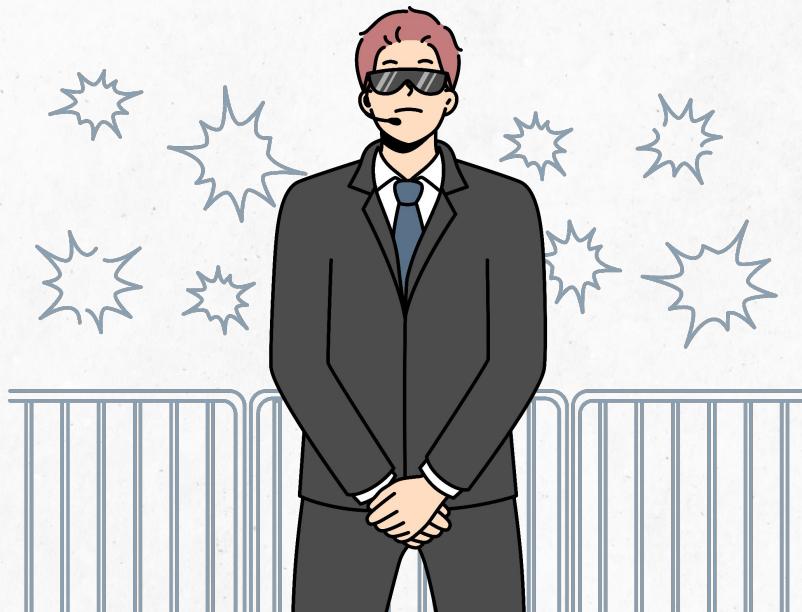
# Bloco 3

---

Explorando os operadores aritméticos e lógicos.

## Vamos analisar esse problema de lógica?

Imagine que você é um guarda em uma ponte que só pode suportar o peso de até duas pessoas por vez. Três pessoas desejam atravessar a ponte, mas elas têm velocidades diferentes para atravessá-la no escuro, ou seja, o tempo que cada uma leva para atravessar é diferente. Além disso, elas só têm uma lanterna, o que significa que alguém precisa voltar com a lanterna para que os outros possam atravessar.



# Vamos analisar esse problema de lógica?

As três pessoas são:

Alice, que atravessa a ponte em um minuto;

Bob, que atravessa a ponte em dois minutos;

Carol, que atravessa a ponte em cinco minutos.



## Como seria a variável?

O problema é encontrar a maneira mais eficiente de fazer com que todas as três pessoas atravessem a ponte em tempo mínimo. Como ajudá-las a fazer isso?

Confira a seguir!



# Operadores lógicos

Os operadores lógicos em JavaScript são usados para realizar operações lógicas em valores booleanos (verdadeiro ou falso) ou expressões condicionais.

Eles permitem que você crie condições mais complexas com base em condições simples.

Existem três operadores lógicos principais em JavaScript:

**E lógico (&&)**

**OU lógico (||)**

**NÃO lógico (!)**

# E lógico (**&&**)

O operador **&&** retorna **true** se ambos os operandos forem verdadeiros.

Além disso, retorna **false** se pelo menos um deles for falso.

## Exemplo

```
var temIdadeParaDirigir = true;  
var temCarteiraDeMotorista = true;  
var podeDirigir = temIdadeParaDirigir && temCarteiraDeMotorista;  
// A variável será verdadeira, pois ambos os operandos são verdadeiros.
```



# OU lógico (||)

O operador `||` retorna **true** se pelo menos um dos operandos for verdadeiro e **false** se ambos forem falsos.

## Exemplo

```
var temConvite = false;  
var temIngresso = true;  
var podeEntrarNaFesta = temConvite || temIngresso;  
// A variável será verdadeira, pois pelo menos um operando é verdadeiro.
```



# NÃO lógico (!)

O operador `!` inverte o valor de um operando booleano. Se o operando for *true*, ele se torna *false*, e se for *false*, se torna *true*.

## Exemplo

```
var choveHoje = false;  
var naoChoveHoje = !choveHoje;  
// A variável será verdadeira, pois invertemos o valor de "choveHoje".
```



► □ ○ ▶ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷ □ ○ ▷

## Depois do que você aprendeu, como solucionar o caso?



Dê palpites!

# Bloco 4

---

Explorando os operadores de *strings*,  
concatenados e ternários.

# Quem sou eu?

Analise as afirmações abaixo e identifique a qual tipo de operador lógico cada uma se refere.

Imagine que você está criando um sistema de segurança para uma porta de alta tecnologia. A porta se abre se houver um cartão autorizado e uma senha.

Suponha que você esteja criando um sistema de compra *on-line* e deseja aplicar descontos com base em certas condições. O cliente recebe um desconto se for assinante *prime* ou se o valor total da compra for superior a R\$ 200,00.

Suponha que você esteja desenvolvendo um sistema de cadastro de eventos e deseja aplicar algumas regras de elegibilidade. O participante não pode se inscrever se não atender à idade mínima.

# Gabarito

E aí, acertou?

Imagine que você está criando um sistema de segurança para uma porta de alta tecnologia. A porta se abre se houver um cartão autorizado e uma senha.

E lógico

Suponha que você esteja criando um sistema de compra *on-line* e deseja aplicar descontos com base em certas condições. O cliente recebe um desconto se for assinante *prime* ou se o valor total da compra for superior a R\$ 200,00.

OU lógico

Suponha que você esteja desenvolvendo um sistema de cadastro de eventos e deseja aplicar algumas regras de elegibilidade. O participante não pode se inscrever se não atender à idade mínima.

NÃO lógico

## Operador de *string*

É um tipo de dado em programação que representa uma sequência de caracteres, que podem ser letras, números, símbolos, espaços em branco ou outros caracteres especiais.

São frequentemente usados para representar texto em programas.



# Como as *strings* são representadas?

As *strings* são delimitadas por aspas duplas (" "). Alguns exemplos podem ser:

“Olá, mundo!”

“12345”

“Nome do usuário”

“Sobrenome”

# Concatenação (+)

É o processo de combinar duas ou mais *strings* para criar uma nova *string* mais longa, usando o símbolo de adição (+).

## Exemplo

```
nome = "John"  
sobrenome = "Doe"  
nome_completo = nome + " " + sobrenome  
print(nome_completo) # Isso irá imprimir "John Doe"
```

# Operador ternário

Também conhecido como operador condicional, é uma construção de controle de fluxo que permite avaliar uma expressão condicional e retornar um valor com base nessa condição.

É nomeado de **ternário** porque possui três partes:

1

A condição a ser avaliada;

2

O valor a ser retornado se a condição for verdadeira;

3

O valor a ser retornado se a condição for falsa.

# Como os operadores ternários são avaliados?

Sua sintaxe é representada por:

```
resultado = (condição) ? valor_se_verdadeiro : valor_se_falso
```

Analise o exemplo:

```
var numero = 10;  
var parOuImpar = (numero % 2 === 0) ? "Par" : "Ímpar";  
console.log(parOuImpar); // Isso irá imprimir "Par"
```

# Bloco 5

---

Investigando os operadores de tipos, os  
relacionais e os de precedência.

# Explique para um famoso

## Vamos realizar uma dinâmica?

Nesta atividade, o desafio é explicar o que é uma *string*, uma concatenação e um operador ternário para um famoso. A pessoa da vez é a **Dory**.

Lembre-se: ela tem memória curta.



# Operador de tipos

O operador **typeof** é usado em JavaScript para determinar o tipo de dado de uma variável ou expressão. Ele retorna uma *string* que representa o tipo de dado da variável. Aqui estão alguns exemplos:

typeof "John"	Retorna "string"
typeof 3.14	Retorna "number"
typeof NaN	Retorna "number"
typeof false	Retorna "boolean"
typeof [1,2,3,4]	Retorna "object"

# Operadores relacionais

São usados para comparar dois valores e retornar um valor booleano que indica se a comparação é verdadeira ou falsa. Um exemplo pode ser:

```
5 > 3 // Retorna true
```

```
10 < 5 // Retorna false
```

```
7 == "7" // Retorna true (comparação não estrita, os valores são convertidos antes  
da comparação)
```

# Precedência

Determina a ordem na qual as operações são realizadas em uma expressão. Operadores com maior precedência são avaliados antes dos operadores com menor precedência.

- Multiplicação e divisão têm uma precedência mais alta do que adição e subtração, então  $6 + 6 / 2$  é avaliado como  $(6 + (6 / 2))$ , resultando em 9;
- Parênteses podem ser usados para alterar a precedência. Por exemplo,  $(6 + 6) / 2$  resultaria em 6;
- Operadores relacionais e lógicos também têm precedência, porém, operadores lógicos têm precedência mais baixa do que os relacionais.

# Ordem de precedência dos operadores

Vamos conhecer a ordem de precedência de operadores em JavaScript!

## Operadores aritméticos

- “( )”: parênteses
- “\*\*”: exponenciação
- “\*”, “/” e “%”: multiplicação, divisão e resto da divisão
- “-” e “+”: subtração e adição

## Operadores relacionais

Não possuem ordem de precedência.

## Operadores lógicos

! - NÃO  
&& - E  
|| - OU

# Bloco 6

---

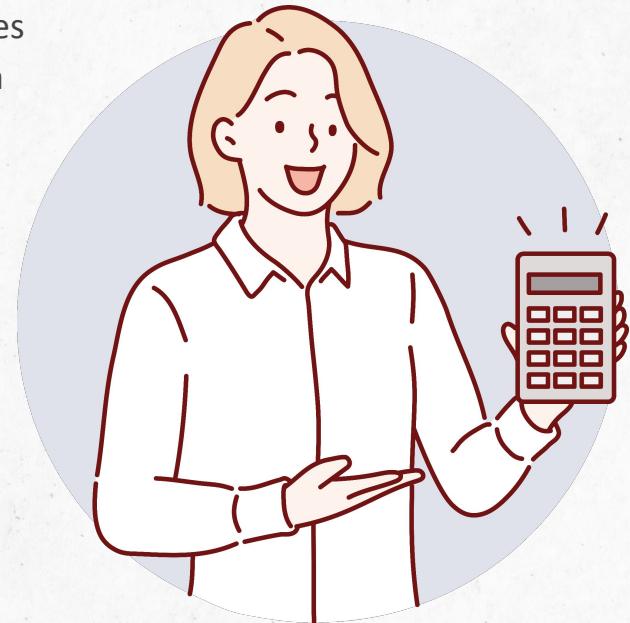
Hora de praticar!

# Vamos praticar tudo o que vimos até aqui?

Separem-se em grupos e distribuem-se de acordo com os computadores ou *notebooks* disponíveis. Depois, vocês precisam acessar a plataforma code.org.

## O que é preciso fazer?

Com o auxílio do professor, vocês devem criar um aplicativo de uma calculadora cujo objetivo é calcular o desconto sobre o preço dos produtos.



# É hora de discutir!

Qual foi a sua maior dificuldade?

O que você sente que aprendeu?

O que você acha que precisa explorar mais?



# Referências Bibliográficas

PROZ EDUCAÇÃO. *Desenvolvimento para Dispositivos Móveis I*. 2023.