

# JAVASCRIPT - MÓDULO 1 - AULA 1

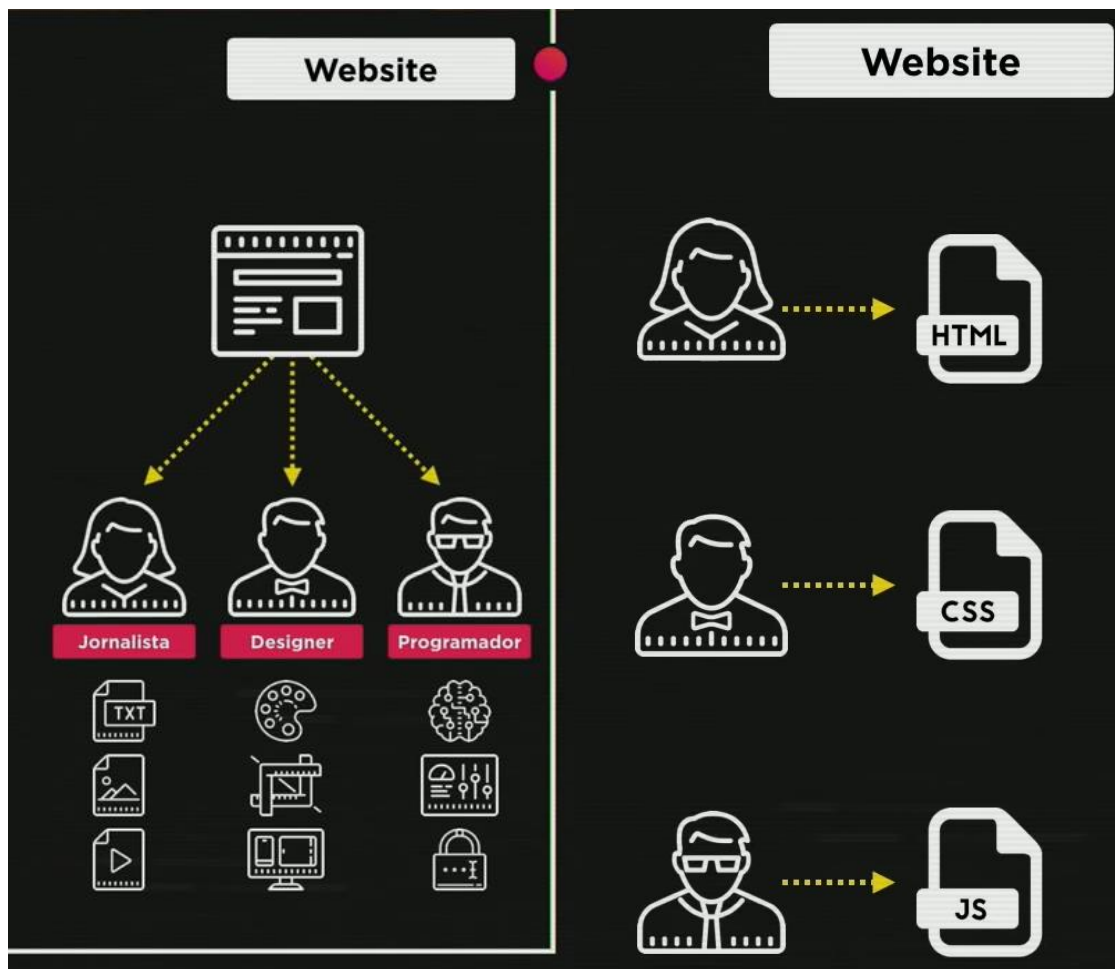
## 1. Cliente x Servidor

Quando estamos utilizando a internet, independente do uso que escolhemos fazer, estamos exercendo o papel de **CLIENTE**, isto é, eu cliente estou utilizando a internet para acessar por exemplo um e-mail. Mas, para acessar a minha conta de e-mail existe um lugar que contém todas as minhas informações para que eu possa visualizá-las ao logar em minha conta, este local é o **SERVIDOR**. É nele que fica tudo armazenado, para que qualquer cliente possa acessar sem maiores problemas.

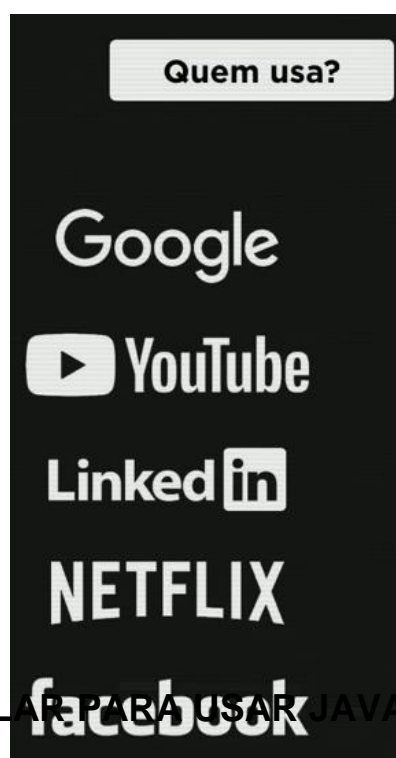


## 2. O que o JavaScript faz?

Como vimos nas aulas de HTML e CSS, para criarmos um site, primeiro precisamos adicionar o CONTEÚDO(HTML) e posteriormente o DESIGN(CSS). Mas resta uma parte para que tenhamos um site completo, a INTERAÇÃO(JavaScript). Abaixo podemos ver um exemplo do uso dessas três vertentes simulando uma empresa em que temos uma jornalista, um design e um programador.



Como pudemos observar no exemplo acima cada um tem seu papel importante na hora de desenvolver um site.



### 3. O QUE INSTALAR PARA USAR JAVASCRIPT?

Para a utilização do JavaScript, devemos ter um compilador do código JS (que no caso estamos já utilizando o Visual Studio).

## 4. MODOS DE USAR JAVASCRIPT

Existem maneiras diferentes de trabalhar com o JavaScript em conjunto com o HTML e CSS. Podemos utilizar o modo **INTERNO** ou o modo **EXTERNO**. No modo interno temos a opção de trabalhar com os comandos JavaScript na mesma página em que é feito o HTML. Isso faz com que fique mais próximo a execução e entendimento dos comandos, porém também consumirá boa parte do arquivo HTML com comandos JavaScript (até porque veremos que alguns sites possuem mais comandos JS do que HTML). Da outra maneira é criado um link de direcionamento para um arquivo .js externo, no qual não ocupará o mesmo arquivo do HTML. É bom pois fica mais organizado (visualmente falando), só devemos nos atentar a linkar corretamente para que a página não tenha problemas.

### Exemplo INTERNO

```
1 <!DOCTYPE html>
2 <html lang="pt-br">
3 <head>
4   <meta charset="UTF-8">
5   <title>HTML + JS interno</title>
6 </head>
7 <body>
8   <h1>Este é o HTML com JS interno</h1>
9   <script>
10    window.alert("Este é um documento
11    HTML com JavaScript interno!")
12  </script>
13 </body>
14 </html>
```

Repare que o JavaScript interno tem como principal característica o uso da **TAG SCRIPT** dentro do **BODY**. E portanto toda a codificação JS fica dentro da TAG.

### Exemplo EXTERNO

```
1 <!DOCTYPE html>
2 <html lang="pt-br">
3 <head>
4   <meta charset="UTF-8">
5   <title>HTML + JS Externo</title>
6   <script src="script.js"></script>
7 </head>
8 <body>
9   <h1>Este é o HTML com JS Externo</h1>
10 </body>
```

```
1 window.alert("Este é um exemplo JS
   externo");
```

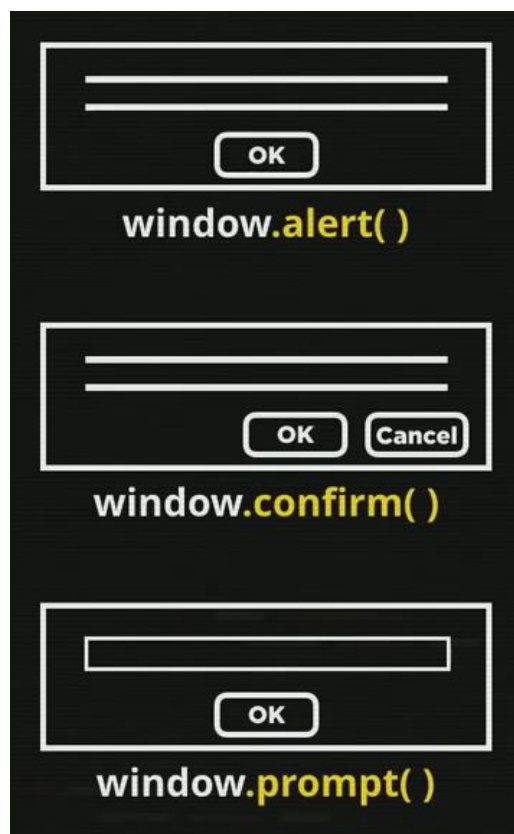
## 5. CONSOLE

Dentre umas das funções iniciais do Javascript temos o **Console**, lá conseguimos testar alguns códigos para começar a entender o funcionamento do Javascript.

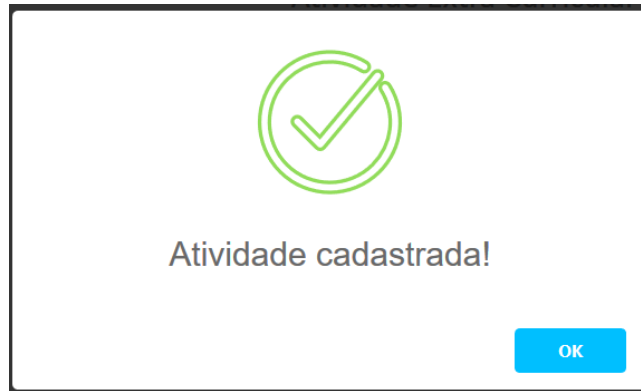
```
console.log("Teste")
```

## 6. JANELAS

Dentre umas das funções iniciais do JavaScript (e principais) temos as **Janelas**. Temos diversas janelas na programação em JS mas são três as mais utilizadas são elas: alert, confirm e prompt. Em vários comandos no JS observaremos que temos a opção de usar o comando em sua maneira simples ou completa. No comando de Janela podemos usar com ou sem a palavra **WINDOW**



- 6.1. Alert:** Esta janela tem a função de trazer uma informação para alertar o usuário de algo que foi concluído, executado, ou até mesmo de algo que o usuário deve tomar atenção.



Utilizando o comando:

```
1  <!DOCTYPE html>
2  <html lang="pt-br">
3  <head>
4  |   <meta charset="UTF-8">
5  |   <title>Janelas</title>
6  </head>
7  <body>
8  |   <h1>Assim se usa a Janela Alert</h1>
9  |   <script>
10 |       window.alert("Olá mundo!")
11 |       alert("Olá mundo")
12 |   </script>
13 </body>
14 </html>
```

Como podemos observar, o comando contém a sintaxe **window.alert()** e dentro das aspas incluiremos uma informação em **TEXTO**. (Pode ser tanto em aspas simples como aspas duplas).

- 6.2. Confirm:** Esta janela tem a função de trazer uma condição que pode ser verdadeira ou falsa, dependendo do que o usuário clicar. Se clicar na opção sim (pode ser ok, concluir, etc) resultará em uma ação, agora se clicar em não (ou cancelar, negar, etc) resultará em uma outra ação.



## Tem a certeza?

Se eliminar não voltará a ver este conteúdo!

Cancel

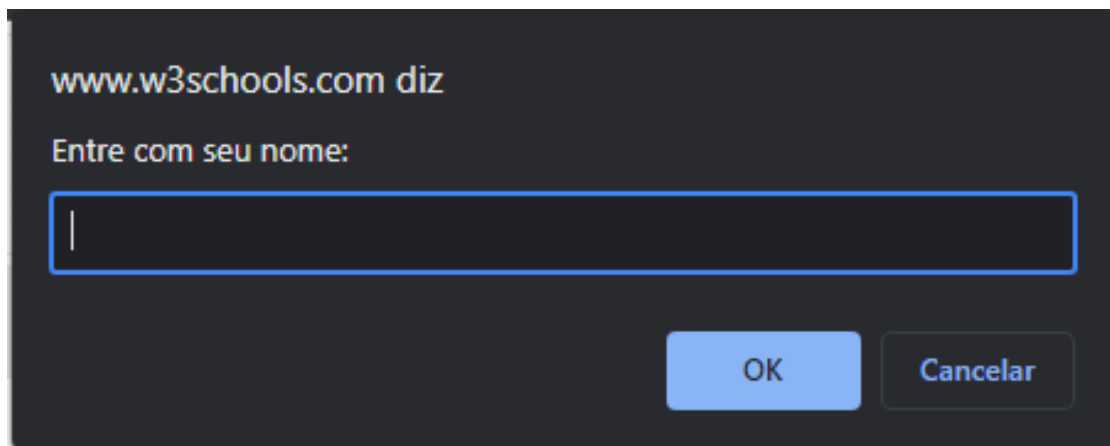
Sim, apagar isto!

Utilizando o comando:

```
1  <!DOCTYPE html>
2  <html lang="pt-br">
3  <head>
4  |   <meta charset="UTF-8">
5  |   <title>Janelas</title>
6  </head>
7  <body>
8  |   <h1>Assim se usa a Janela Confirm</h1>
9  |   <script>
10 |       window.confirm("Olá, clique em ok ou cancelar")
11 |       confirm("Olá, clique em ok ou cancelar")
12 |   </script>
13 </body>
14 </html>
```

Como podemos observar, o comando contém a sintaxe **window.confirm(" ")** e dentro das aspas incluiremos uma informação em **TEXTO**. (Pode ser tanto em aspas simples como aspas duplas). Esse é o exemplo mais simples da utilização da Janela confirm. Aprenderemos novas maneiras de estilizar e modificar essa janela assim que aprendermos os comandos condicionais.

- 6.3. Prompt:** Esta janela tem a função de receber uma informação vinda do usuário. Seja para preenchimento de alguma informação, confirmação de informações ou até para cálculos no geral.



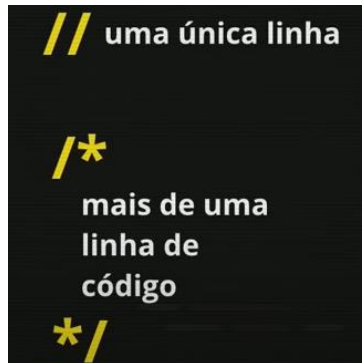
Utilizando o comando:

```
1  <!DOCTYPE html>
2  <html lang="pt-br">
3  <head>
4  |   <meta charset="UTF-8">
5  |   <title>Janelas</title>
6  </head>
7  <body>
8  |   <h1>Assim se usa a Janela Prompt</h1>
9  |   <script>
10 |       window.prompt("Olá, quem é você? ")
11 |       prompt("Olá, quem é você? ")
12 |   </script>
13 </body>
14 </html>
```

Como podemos observar, o comando contém a sintaxe **window.prompt(" ")** e dentro das aspas incluiremos uma informação em **TEXTO**. (Pode ser tanto em aspas simples como aspas duplas). Esse é o exemplo mais simples da utilização da Janela prompt. Aprenderemos novas maneiras de estilizar e modificar essa janela assim que aprendermos os comandos condicionais.

## 7. COMENTÁRIOS

Os comentários têm um papel muito importante de informar, explicar ou dar instruções dentro de um código, tanto em linguagem de programação quanto de marcação. No JS não é diferente, portanto a seguir temos os métodos de utilização dos comentários na linguagem JavaScript.



### 7.1. Em código

```
1 <!DOCTYPE html>
2 <html lang="pt-br">
3 <head>
4   <meta charset="UTF-8">
5   <title>Comentários</title>
6 </head>
7 <body>
8   <h1>Assim se usa comentários em JS</h1>
9   <script>
10    window.alert("Estamos testando os comentários") // Essa é uma maneira de comentar em JS
11    window.alert("Temos duas maneiras de comentar em JS") /* Essa é a outra maneira de comentar em JS */
12  </script>
13 </body>
14 </html>
```

Como se pode observar, temos duas maneiras de fazer os comentários. Podemos usar tanto uma quanto a outra, ou as duas no mesmo código.

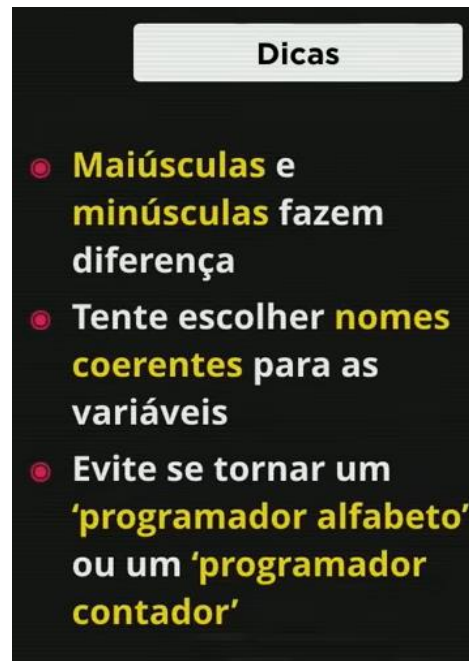
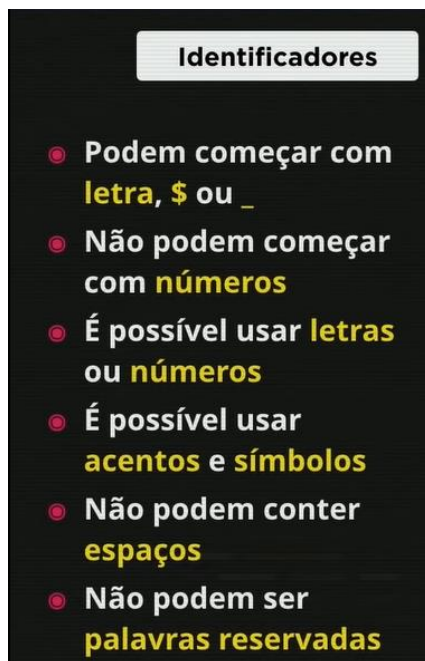
## 8. VARIÁVEIS

Já aprendemos em Python o que é e como funciona uma variável. Porém, em JavaScript temos algumas características um pouco diferentes, além é claro da sintaxe que também é outra.

Dentro do JavaScript temos três maneiras de declarar uma variável. **LET** ou **VAR**. Antes de conhecermos cada um desses três modos, vamos entender dois termos importantes para ao conhecer os modos de declaração não ficar dúvidas.

- **Declaração:** especifica o tipo da variável e o seu nome
- **Atribuição:** é utilizado para armazenar um valor em uma variável





Acima temos alguns critérios que devemos ter ao criar uma variável. Ao criar uma variável devemos seguir atentamente a esses critérios, pois a falta de adesão de qualquer um deles pode resultar em bugs no código, atrapalhando o trabalho do programador.

- 8.1. **LET:** No modo let ao declararmos uma variável ela tem função restrita apenas ao bloco em que está, isto é, é considerada uma variável **LOCAL**. Além disso, ela permite a reatribuição, porém, não permite a redeclaração.
- 8.2. **VAR:** No modo de declaração var, é de utilização **GLOBAL** portanto **não** respeita bloco além de permitir a redeclaração e reatribuição.
- 8.3. **CONST:** O modo de declaração const é **LOCAL** e respeita os blocos. **Não** permitindo reatribuição **nem** redeclaração.
- 8.4. **TIPOS DE VARIÁVEL:** Como é em toda linguagem de programação temos tipos diferentes de variáveis dependendo da necessidade. Os tipos mais comuns são numéricos e de texto. Vamos conhecer agora os principais tipos de variável, e posteriormente ao aprofundar no conteúdo os demais tipos.
  - 8.4.1. **Variáveis numéricas:** Para definir os tipos de variáveis numéricas temos o comando number. Nas variáveis numéricas contamos com dois tipos - **Inteiros e reais** (Respectivamente na imagem abaixo)



- **Declaração:** Abaixo também temos a maneira de definir uma variável como tipo inteira ou real. Você deve colocar a sintaxe **Number.parseInt(conteúdo)** para **números inteiros**, **Number.parseFloat(conteúdo)** para **números reais** ou apenas **Number(conteúdo)** para **ambos os tipos**.

```
Number.parseInt(n)
Number.parseFloat(n)
Number(n)
```

- **Exemplo:** As imagens a seguir mostram como ficaria as três opções de declaração. Int, float e apenas number, respectivamente.

```
var a = 2
var b = 3.75
```

Aqui declaramos quando já temos o valor pré-definido

```
<body>
  <h1>Assim se usa ParseInt no JS</h1>
  <script>
    var m = Number.parseInt(window.prompt("Digite um valor: "))
    var n = Number.parseInt(window.prompt("Digite outro valor: "))
    var s = m + n
    window.alert("A resposta é: " + s)
  </script>
</body>
```

Declaração + recebimento de números inteiros

```
<body>
  <h1>Assim se usa parseFloat no JS</h1>
  <script>
    var m = Number.parseFloat(window.prompt("Digite um valor: "))
    var n = Number.parseFloat(window.prompt("Digite outro valor: "))
    var s = m + n
    window.alert("A resposta é: " + s)
  </script>
</body>
```

**Declaração + recebimento de números reais**

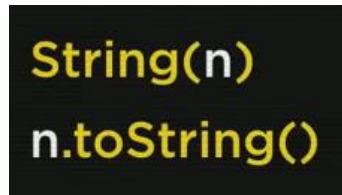
```
<body>
  <h1>Assim se usa apenas o NUMBER no JS</h1>
  <script>
    var m = Number(window.prompt("Digite um valor: "))
    var n = Number(window.prompt("Digite outro valor: "))
    var s = m + n
    window.alert("A resposta é: " + s)
  </script>
</body>
```

**Declaração + recebimento com Number apenas**

- 8.4.2. Variáveis de texto:** O tipo de variável para texto tem o nome praticamente igual em todas as linguagem de programação. Chamado de **STRING** ele é utilizado tanto para receber alguma informação digitada pelo usuário quanto também para orientações / solicitação para receber um valor em uma variável. Abaixo temos a imagem de 3 tipos de declaração para a variável STRING. Vamos entender a diferença entre os tipos nos exemplos.



- **Declaração:** Para declarar uma variável tipo STRING basta digitarmos a sintaxe **String(conteúdo)**. Podemos observar outra opção na imagem abaixo, essa opção pode também ser utilizada, mas vamos entendê-la primeiro. Nesse caso o conteúdo (valor da variável), está fora do parênteses, antes da sintaxe e o parênteses encontra-se vazio. Isto é, nesse caso nós receberemos qualquer valor (não importando o tipo), e vamos **CONVERTÊ-LO** para string. Dentro do parênteses então ficaria uma variável que receberia a transição da variável para string.



The image shows two syntaxes for string conversion. The first is **String(n)** where 'n' is inside the parentheses. The second is **n.toString()** where 'n' is outside and the parentheses are empty.

- **Exemplos:** A seguir vamos validar os exemplos do uso de variáveis do tipo String. Tanto para receber valor, quanto para fazer essa conversão já em outra local de armazenamento (outra variável)

```
<h1>Assim se usa String no JS</h1>
<script>
//declarando valores tipo string
var apelido = "Greg"

//recebendo valores tipo string
var nome = String(window.prompt("Digite seu nome: "))
window.alert("Seu nome é: " + nome)
</script>
```

Declarando ou recebendo um valor STRING

## 8.5. Variáveis Booleanas:



The image shows a dark red rounded rectangle with a dashed white border containing the words 'true' and 'false' in white. Below this rectangle is a smaller, light gray rounded rectangle with the word 'boolean' in black.

- **Declaração:** Uma variável do tipo **Boolean** tem o único papel de definir se o valor que você designou a ela é **VERDADEIRO (true)** OU **FALSO (false)**. Existem duas maneiras de definir uma variável booleana. A primeira delas é chamada de maneira **PRIMITIVA**. Essa aprenderemos agora, a outra é chamada de booleana em objeto onde usaremos em condicionais. Dessa outra maneira veremos mais a frente com o avançar do nosso curso. A imagem

abaixo mostra como é fácil a utilização da variável e que tem uso pouco aparente nos programas, devido a ser muito específico o seu uso.

```
> var m = true
undefined
> m
true
>
```

- **Exemplos:** A seguir um exemplo do uso da variável booleana no modo Primitivo

```
<h1>Assim se usa apenas o Boolean no JS</h1>
<script>

var teste = false
var teste2 = true

</script>
```

Declarando um valor BOOLEAN

## 9. FORMATAÇÃO DE DADOS

A formatação de dados em JavaScript, busca facilitar muito o dia-a-dia do programador. Trazendo sintaxes diretas e simples para que possamos fazer algumas alterações em valores números e de texto. Vamos validar as formatações de ambos.

- 9.1. **Valores numéricos:** Na formatação de valores existem diversas possibilidades, vamos validar duas delas que são muito importantes para o programador ao desenvolver sistemas e sites.

- **toFixed():** Esse comando tem a função de adicionar a quantidade casas decimais que decidir a sua variável. Vejamos a seguir.

```
> var num = 2345.2
undefined
> num
2345.2
> num.toFixed(2)
'2345.20'
```

Como pudemos observar acima, no início criamos a variável com um valor com apenas uma casa decimal. E ao adicionar o comando a variável escolhemos 2 casas decimais. No final o valor já alterado está com as duas casas decimais.

- **toLocaleString:** Esse comando tem por função **transformar** qualquer valor **numérico comum** em **moeda**, basta selecionar qual país deseja a conversão e ele fará a conversão já adaptando o que for preciso.

```
> num.toLocaleString('pt-Br', {style: 'currency', currency: 'BRL'})
'R$ 2.345,20'
> num.toLocaleString('pt-Br', {style: 'currency', currency: 'USD'})
'US$ 2.345,20'
> num.toLocaleString('pt-Br', {style: 'currency', currency: 'EUR'})
'€ 2.345,20'
```

Utilizando a mesma variável do exemplo acima, podemos observar a conversão de moeda para três países diferentes. Mudando apenas a sigla da moeda de cada país.

- 9.2. Strings:** Quando utilizamos uma String, estamos adicionando uma informação de texto. O que não quer dizer que não possamos, por exemplo, adicionar informações de uma variável para agregar a este texto. Na imagem abaixo conseguimos entender como funciona.

```
// Formatando Strings
var s = 'JavaScript'
'Eu estou aprendendo s' // não faz interpolação
'Eu estou aprendendo' + s // usa concatenação
`Eu estou aprendendo ${s}` // usa template string
```

Ao observarmos a imagem, podemos ver que possuem 3 situações. A primeira temos um exemplo que vai resultar em um erro, pois o JavaScript **NÃO FAZ a interpolação** só colocando a variável entre as aspas. Isto é, se colocar uma variável dentro das aspas onde teria um texto, o programa não vai entender que é uma variável e trará apenas o texto.

Na segunda situação é utilizada a **concatenação** da variável para juntá-la com o texto. Dessa maneira sim poderemos ver em conjunto, o texto e a variável, um ao lado do outro.

Já na terceira situação é utilizado o modo **template string**. Diferente do primeiro caso, essa é a maneira correta de utilizar **INTERPOLAÇÃO** no JavaScript. Portanto, caso queira colocar variável entre textos podemos fazer dessa maneira. Essa é uma das maneiras, iremos conhecer outros adiante.

```
var s1 = "Teste de String"
var s2 = 'estou testando'
var s3 = `todos os modos`
```

Outro ponto importante da formatação de strings é entender a diferença das aspas e do acento grave. Quando utilizamos aspas duplas ou aspas simples elas trarão o significado de texto sem **INTERPOLAÇÃO** (como já vimos nas descrições acima). Portanto, quando se usa o acento grave, podemos adicionar o **template string** (serve exclusivamente para isso).

## 10. OPERADORES

Muito semelhante a matemática, os operadores vem para fazer cálculos simples, cálculos esses que usamos no nosso dia a dia. No JavaScript existem todos os operadores principais e até alguns que facilitam cálculos para nós. Vamos conhecer os operadores em sequência.

- 10.1. Aritméticos:** Os operadores aritméticos são utilizados para fazer cálculos matemáticos. Cálculos esses que aprendemos na escola e aqui não é diferente (a não ser a sintaxe). Temos Adição(+), Subtração(-), multiplicação(\*), divisão(/), resto da divisão(%) e ao quadrado (\*\*).

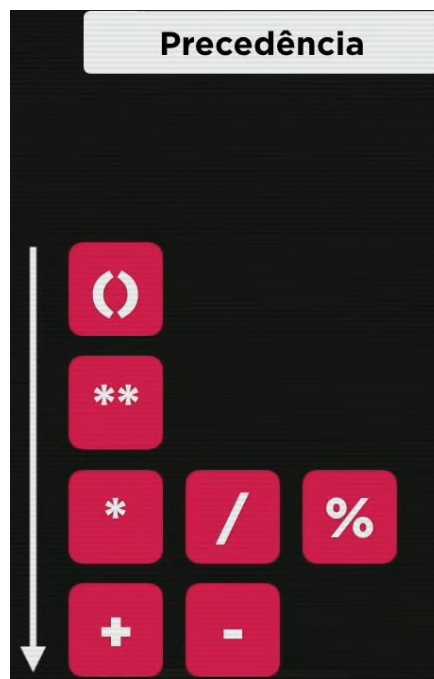


Aritméticos				
5	+	2	→	7
5	-	2	→	3
5	*	2	→	10
5	/	2	→	2.5
5	%	2	→	1
5	**	2	→	25

**ATENÇÃO:** As regras na matemática que usamos no dia a dia se aplica da mesma maneira nas linguagem de programação. Muito cuidado ao fazer contas com adição e divisão por exemplo. Pela regra divisão é feito primeiro, se quiser dar prioridade a uma adição ou subtração por exemplo, adicione **PARÊNTESES**.

**Exemplo:**  $4 + 3 / 3$  (**ERRADO**)

$(4+3) / 3$  (**CORRETO**)





**10.2. Incremento:** Os incrementos são muito comuns nas linguagens de programação. Ele é uma forma resumida de tratar uma variável que tem por função ir acumulando ou subtraindo alguma variável.

**Exemplo:** var **x = 1**

Eu quero somar mais nessa variável x que inicialmente vale 1. Eu posso fazer assim: **x = x+1** (que já é incremento) ou posso fazer de uma maneira resumida: **x+= 1** (isso vale tanto para adição quanto para subtração) ou mais resumida ainda: **x++**

# **Operadores**

**aritméticos**

**atribuição**

**relacionais**

**lógicos**

**ternário**

## Data Types

**number**

**Infinity**

**NaN**

**string**

**boolean**

**null**

**undefined**

**object**

**Array**

**function**

```
> nome = "Marlon"
'Marlon'
> idade = 27
27
> nota = 9
9
> 'O aluno ' + nome + ' com ' + idade + ' anos, tirou nota: ' + nota
'O aluno Marlon com 27 anos, tirou nota: 9'
> 'O aluno ${nome} com ${idade} anos tirou nota ${nota}'
'O aluno ${nome} com ${idade} anos tirou nota ${nota}'
> `O aluno ${nome} com ${idade} anos tirou nota ${nota}`
'O aluno Marlon com 27 anos tirou nota 9'
```