

# Desenvolvimento Web

## JavaScript

ANOS

# JavaScript

---

No início, a World Wide Web (nossa internet) era apenas um grande aglomerado de páginas HTML com links que apontavam uns para os outros e nada mais. Com o passar dos anos, as necessidades de quem navegava na internet foram ficando cada vez mais complexas e exigiam uma forma mais avançada das páginas web interagirem com os navegadores e seus usuários.

Hoje, a realidade é completamente diferente. A internet não é mais composta por meros documentos HTML com um punhado de texto e imagens, mas sim por aplicações completas e funcionais que facilitam enormemente o dia-a-dia de todos. E tudo isso graças ao surgimento de uma certa tecnologia que está presente em nossa vida digital, mesmo que sequer nos demos conta disso: o [JavaScript](#).

JavaScript é uma linguagem de programação criada em 1995 por Brendan Eich enquanto trabalhava na **Netscape Communications Corporation**. Originalmente projetada para rodar no Netscape Navigator, ela tinha o propósito de oferecer aos desenvolvedores formas de tornar determinados processos de páginas web mais dinâmicos, tornando seu uso mais agradável. Um ano depois de seu lançamento, a [Microsoft](#) portou a linguagem para seu navegador, o que ajudou a consolidar a linguagem e torná-la uma das tecnologias mais importantes e utilizadas na internet.

# JavaScript

---

Embora ela tenha esse nome, não se deve confundir JavaScript com Java, linguagem de programação desenvolvida pela Sun Microsystems: antes, a linguagem criada pela Netscape recebera nomes como LiveScript e Mocha, mas, para aproveitar o grande sucesso da linguagem da Sun no mercado, os executivos da Netscape resolveram mudar o nome de sua linguagem para o atual. Entretanto, Java e Java Script são completamente diferentes e possuem propósitos diversos.

Mas como o JavaScript funciona? **Ao invés de rodar remotamente em servidores na internet, o JavaScript tem como característica rodar programas localmente - do lado do cliente**, como se costuma dizer em TI. Assim sendo, o JavaScript fornece às páginas web a possibilidade de programação, transformação e processamento de dados enviados e recebidos, interagindo com a marcação e exibição de conteúdo da linguagem HTML e com a estilização desse conteúdo proporcionada pelo CSS nessas páginas.

## JavaScript hoje

Com o grande sucesso do JavaScript, tal tecnologia evoluiu para atender às mais diversas demandas que surgiam com a evolução da internet. Atualmente, é possível não apenas desenvolver sites e aplicativos ricos, mas também aplicativos para smartphones e até mesmo programas desktop. Conheça agora algumas tecnologias que surgiram com a evolução do JavaScript.

# JavaScript

---



## Jquery

A mais famosa biblioteca JavaScript do mercado fornece uma variação dessa linguagem com uma sintaxe mais amigável, o que simplifica a criação de aplicações. Graças ao jQuery, é possível escrever programas em JavaScript mais facilmente, pois a sintaxe original do JavaScript não é tão fácil de aprender.

O jQuery se tornou tão popular que, em muitos casos, desenvolvedores substituem totalmente o JavaScript escrito de forma nativa pelo jQuery para criar suas aplicações. Muitos dos componentes dinâmicos que você está vendo nas páginas do Canaltech foram criados graças a esta biblioteca.

# JavaScript

---



## Nodejs

Embora originalmente o JavaScript tenha sido projetado para rodar em navegadores, atualmente também é possível executar aplicações escritas nessa linguagem em servidores web graças ao Node.js.

Criado em 2009, o Node.js é um conjunto de ferramentas open-source que permite criar servidores web para execução remota de aplicações JavaScript. Serviços importantes como [PayPal](#), [LinkedIn](#) e Groupon usam Node.js para funcionar.

Graças aos avanços proporcionados pela comunidade de desenvolvedores dessa ferramenta, existem implementações do Node.js até mesmo para dispositivos da chamada Internet das Coisas: o Tessel, computador semelhante ao Arduino, executa aplicações embarcadas rodando em Node.js.

# JavaScript

---



## PhoneGap

O JavaScript já transcendeu os navegadores e agora permite fazer virtualmente qualquer software que se possa imaginar, inclusive aplicativos para smartphones!

Todos os sistemas operacionais móveis disponíveis no mercado suportam JavaScript, sendo possível construir aplicativos para Android, iOS, Windows Phone. A vantagem aqui é que geralmente é mais fácil desenvolver aplicativos compatíveis com todas as aplicações, ao contrário do desenvolvimento com linguagens nativas, que limita as opções ao SO de origem (Java para Android, Swift para iOS, etc).

Sencha Touch, PhoneGap, Titanium e outras tecnologias são apenas alguns exemplos de ferramentas que permitem a criação de poderosos aplicativos mobile com mais facilidade e flexibilidade.

# JavaScript

---



## Sublime Text 3

Vamos utilizar o IDE Sublime Text para desenvolver nossas atividades de javascript. Essa IDE é rápida e leve, não exigindo máquinas robustas para o desenvolvimento dos códigos.

Site oficial para fazer o download da IDE  
<https://www.sublimetext.com/>

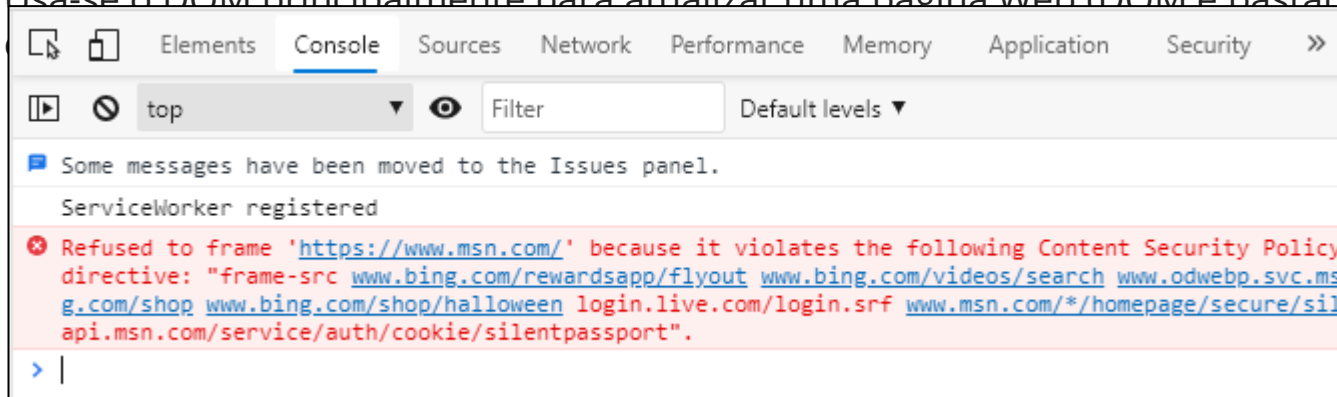
# JavaScript

## DOM

O **Document Object Model** ou simplesmente DOM é utilizado pelo navegador Web para representar a sua página Web. Quando altera-se esse modelo com o uso do Javascript altera-se também a página Web. É muito mais fácil trabalhar com DOM do que diretamente com código HTML ou CSS.

Um dos grandes responsáveis por isso tudo é o **objeto document** que é responsável por conceder ao código Javascript todo o acesso a **árvore DOM do navegador Web**. Portanto, qualquer coisa criado pelo navegador Web no modelo da página Web poderá ser acessado através do objeto Javascript document.

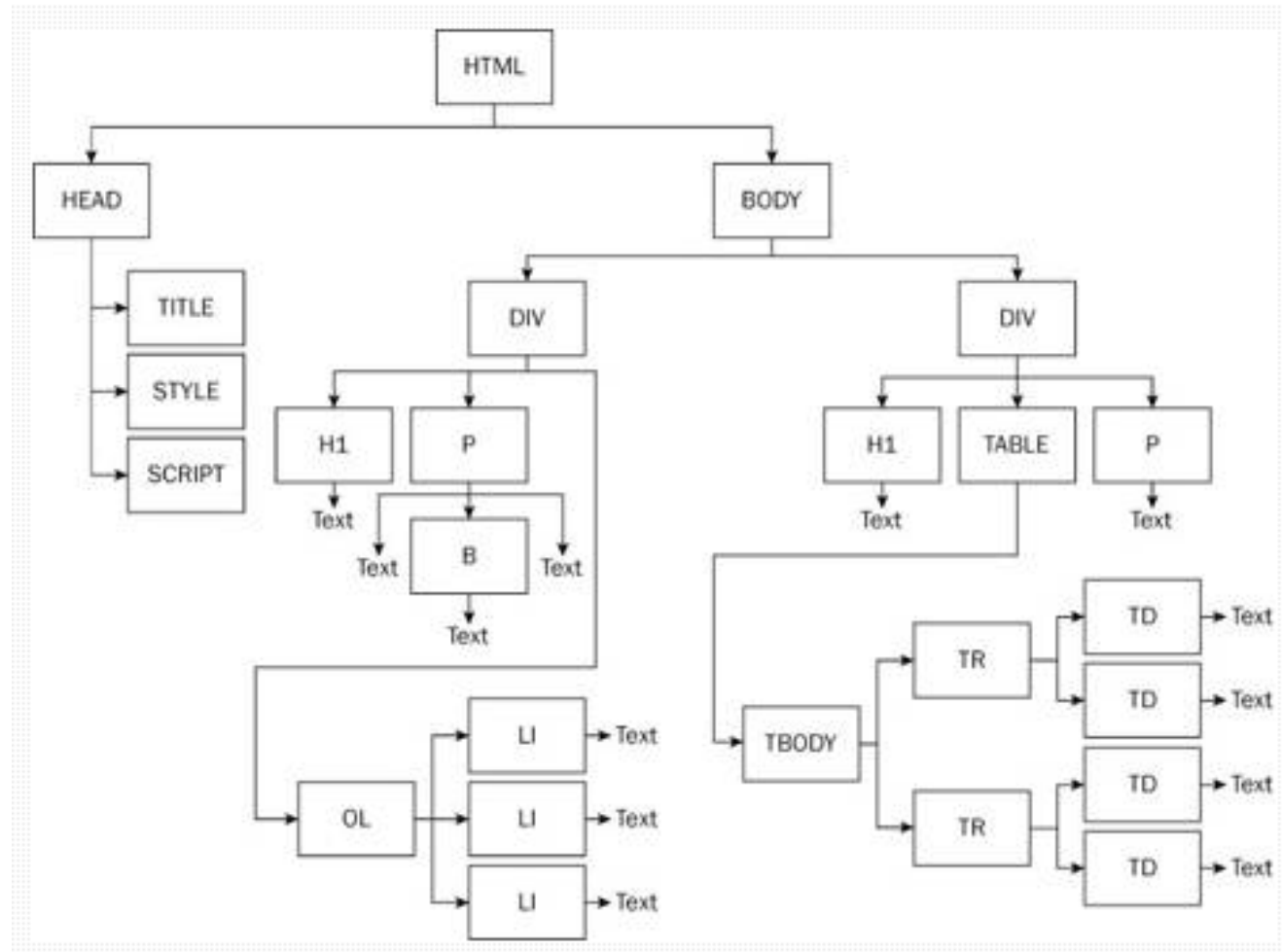
Usa-se o DOM principalmente para atualizar uma página Web (DOM é bastante utilizado com Ajax) **Para o DOM aperte F12.**





# JavaScript

## Árvore do DOM



# JavaScript: Incluindo JavaScript no HTML

---

## Inclusão Interna e Externa

Os códigos JavaScript podem ser inseridos em uma página HTML de duas formas:

- **Internamente:** Criando uma área delimitada pelas tags `<script></script>`
- **Externamente:** É criado um arquivo `.js` e este arquivo é incorporado a página html, assim como fazemos com as folhas de estilo CSS.  
Para isso utilizamos a tag `<script src="caminho do arquivo"></script>`

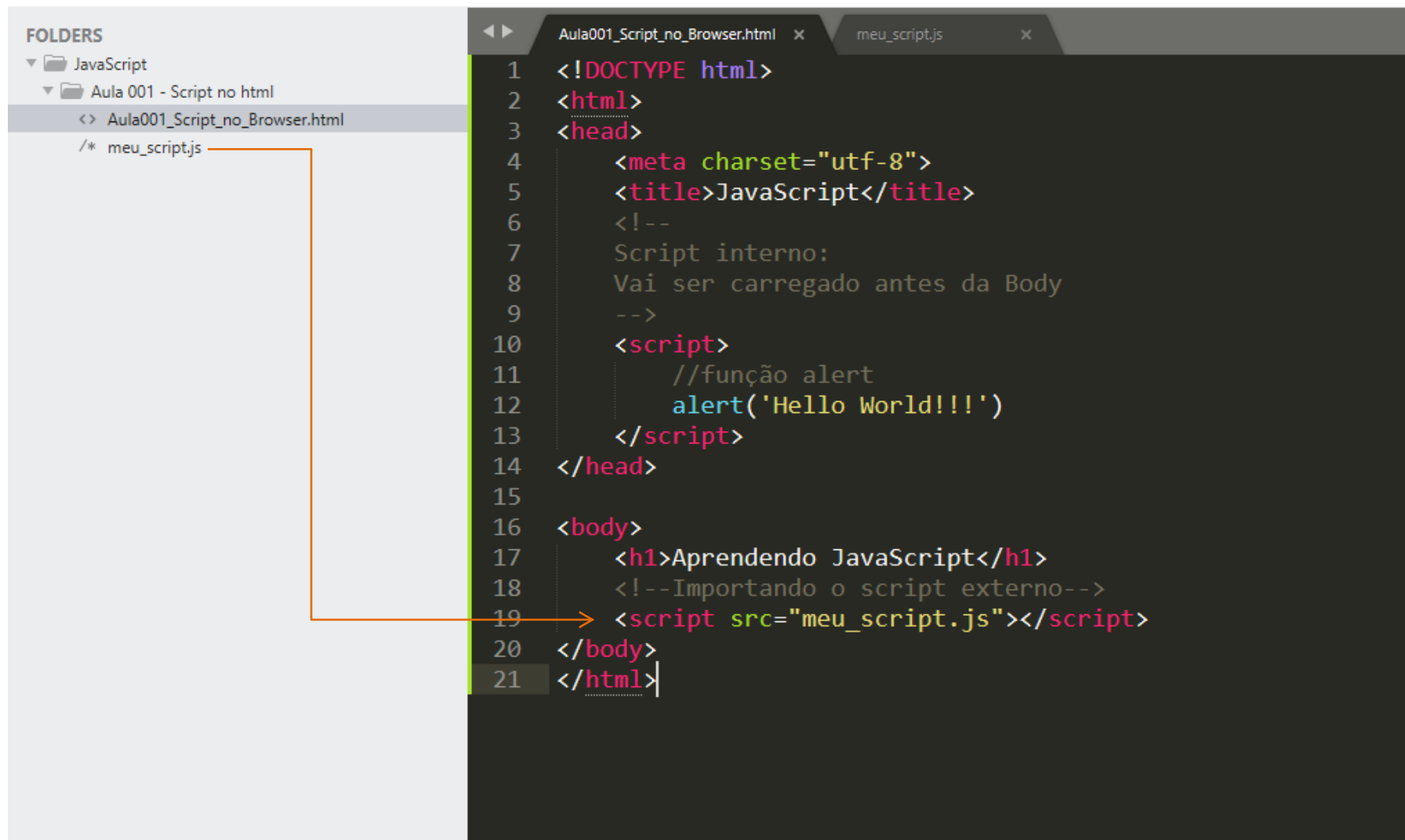
Para essa segunda opção devemos seguir algumas regras:

- O nome do arquivo não pode ter caracteres especiais
- O nome do arquivo não pode conter espaços

## Quando usar interno e quando usar externo?

Fácil! Se o seu código for razoavelmente simples, utilize a primeira opção. Caso contrário, utilize a segunda opção.

# JavaScript: Incluindo JavaScript no HTML



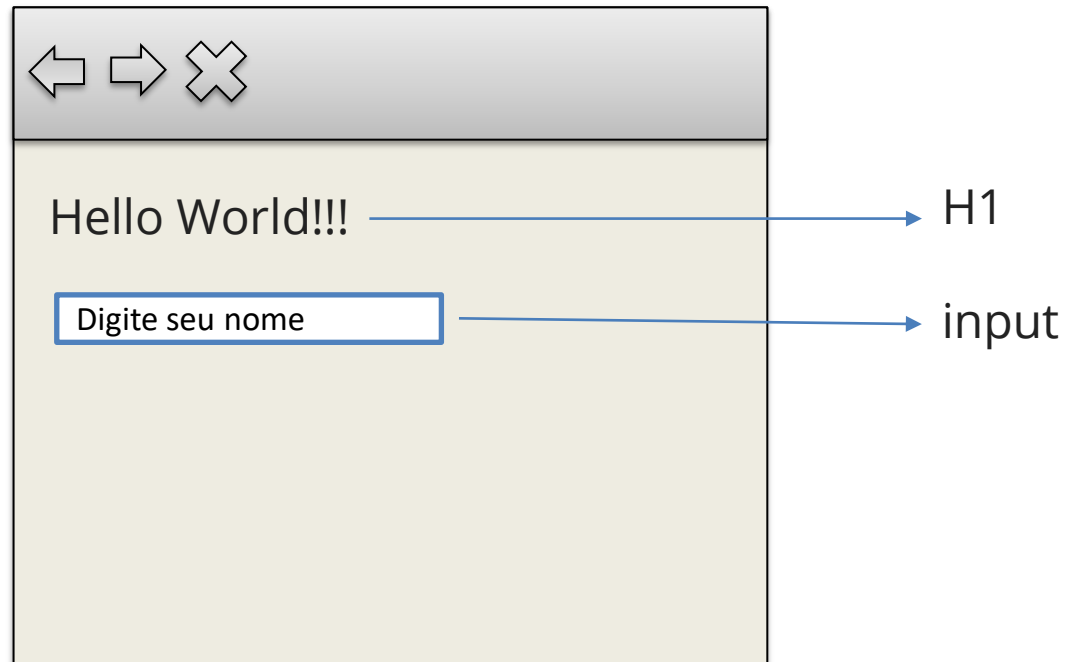
The image shows a code editor interface with two panels. The left panel, titled 'FOLDERS', displays a directory structure: 'JavaScript' (expanded) contains 'Aula 001 - Script no html', which in turn contains 'Aula001\_Script\_no\_Browser.html'. Below this, the file '/\* meu\_script.js' is listed. An orange line originates from this file name and points to the corresponding script tag in the main editor. The main editor has two tabs: 'Aula001\_Script\_no\_Browser.html' (active) and 'meu\_script.js'. The active tab contains the following HTML code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <title>JavaScript</title>
6     <!--
7     Script interno:
8     Vai ser carregado antes da Body
9     -->
10    <script>
11        //função alert
12        alert('Hello World!!!')
13    </script>
14 </head>
15
16 <body>
17     <h1>Aprendendo JavaScript</h1>
18     <!--Importando o script externo-->
19     <script src="meu_script.js"></script>
20 </body>
21 </html>
```

# JavaScript: Ordem de precedência do script

No início da codificação de linguagens interpretadas é muito comum apontarmos para elementos que ainda não foram renderizados na tela causando assim erros inesperados. Por isso tenha em mente quais elementos devem ser inicializados ou tratados antes ou depois do Body.

**Para um melhor entendimento veja o seguinte caso: Se eu quiser tratar o elemento input ele precisará ser criado antes da entrada do script.**

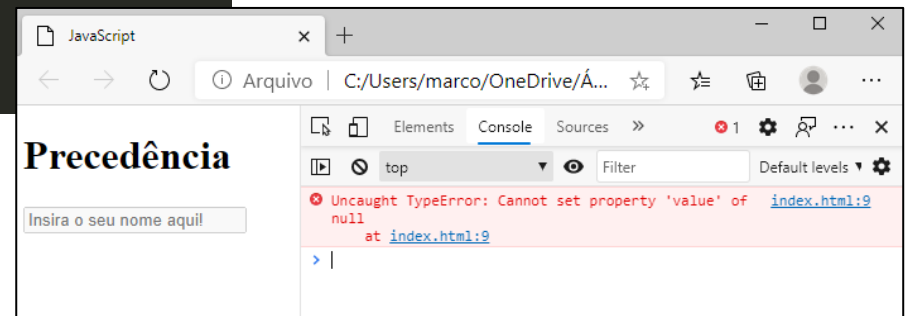


# JavaScript: Ordem de precedência do script

Vai gerar um erro interno, veja com F12.

```
index.html x
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>JavaScript</title>
6
7     <script>
8       //Recuperar o elemento do DOM
9       document.getElementById('nome').value = 'Novo valor!'
10    </script>
11  </head>
12
13  <body>
14    <h1>Precedência</h1>
15
16    <!--Inserindo um elemento Input-->
17    <input
18      type="text"
19      placeholder="Insira o seu nome aqui!"
20      id="nome"
21      disabled="disabled">
22  </body>
23
24 </html>
```

Estou tentando manipular  
um elemento (id) antes de  
renderizar o body



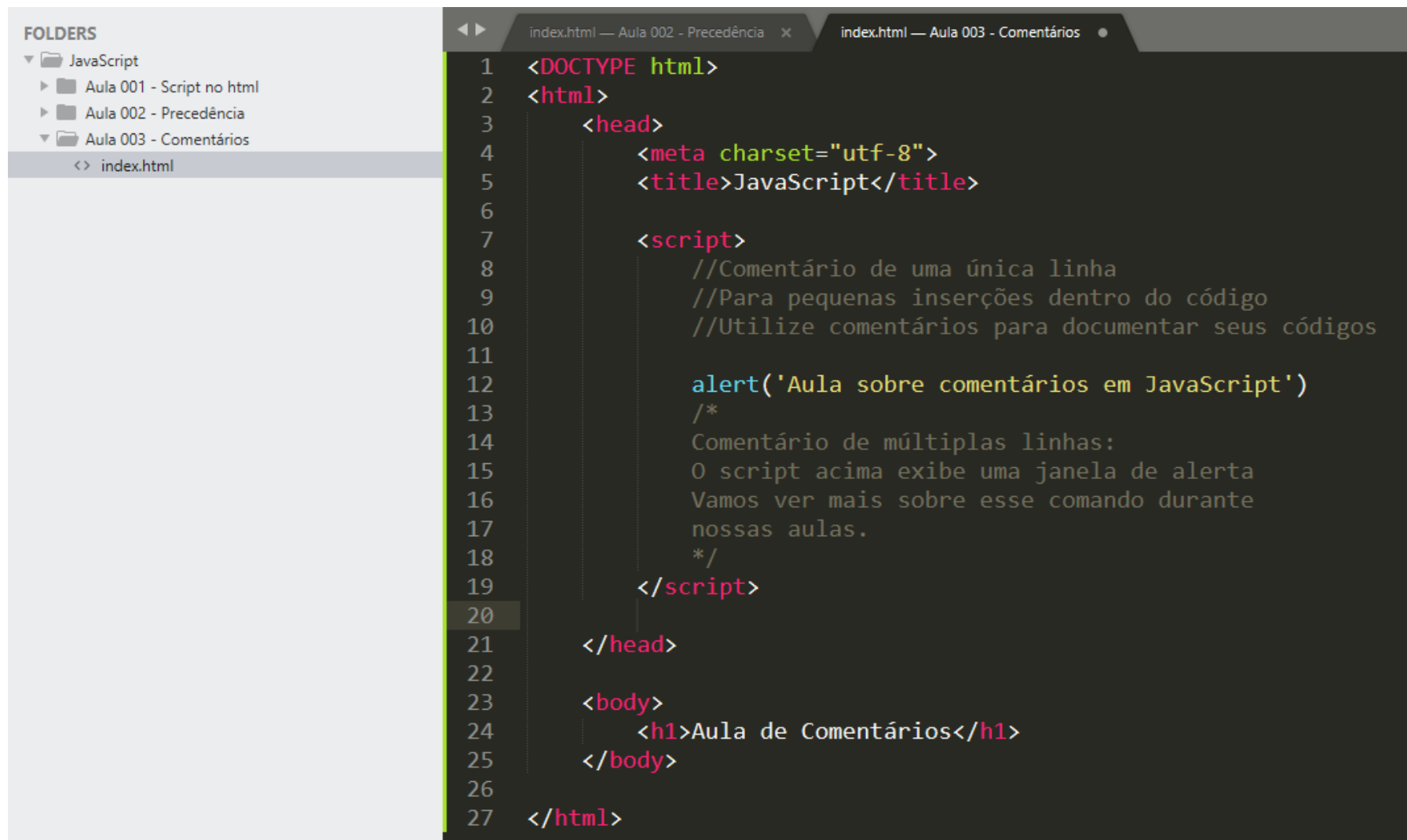
# JavaScript: Ordem de precedência do script

Agora vai funcionar corretamente

```
index.html x
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>JavaScript</title>
6   </head>
7
8   <body>
9     <h1>Precedência</h1>
10
11     <!--Inserindo um elemento Input-->
12     <input
13       type="text"
14       placeholder="Insira o seu nome aqui!"
15       id="nome"
16       disabled="disabled">
17
18     <!--Um local mais coerente nesse caso-->
19     <script>
20       //Recuperar o elemento do DOM
21       document.getElementById('nome').value = 'Novo valor!'
22     </script>
23   </body>
24
25 </html>
```



# JavaScript: Comentários



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a folder named 'JavaScript' containing three subfolders: 'Aula 001 - Script no html', 'Aula 002 - Precedência', and 'Aula 003 - Comentários'. The 'Aula 003 - Comentários' folder is selected, and the file 'index.html' is open. The code editor shows the following HTML code:

```
1 <DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>JavaScript</title>
6
7     <script>
8       //Comentário de uma única linha
9       //Para pequenas inserções dentro do código
10      //Utilize comentários para documentar seus códigos
11
12      alert('Aula sobre comentários em JavaScript')
13      /*
14      Comentário de múltiplas linhas:
15      O script acima exibe uma janela de alerta
16      Vamos ver mais sobre esse comando durante
17      nossas aulas.
18      */
19    </script>
20
21  </head>
22
23  <body>
24    <h1>Aula de Comentários</h1>
25  </body>
26
27 </html>
```

# JavaScript: Variáveis

---

Variáveis seus espaços virtuais onde guardamos informações. É muito comum quando estamos começando a estudar JavaScript declaramos nossas variáveis com o comando `var`. Apesar da forma explícita se opcional é altamente recomendado! Nas nossas aulas e para um processo de desenvolvimento mais moderno utilizaremos o comando `let`. Se habituem a utilizar esse comando em seus códigos daqui para frente. Outras formas modernas de codificação serão tratadas mais à frente.

Tipos:

## **String**

Caracteres e textos:

## **Number** (int e Float)

Int: Números inteiros positivos e negativos

Float: números fracionários positivos e negativos

## **Boolean** (True ou False)

True: Verdadeiro

False: Falso



# JavaScript: Variáveis

---

Para declaramos variáveis em JavaScript precisamos seguir algumas regras:

Declarações:

- Não podem começar com números, apenas letras e "\_".
- Não podem começar com caracteres especiais: "%", "~", "^"
- Não podem ser utilizadas as palavras reservadas da linguagem: var, string, int

Exemplos corretos

```
let nome... let data_nascimento... let numero... let minhaIdade
```

Exemplos incorretos

```
let lnome... let %data_nascimento... let número... let Minha Idade
```

JavaScript também é uma linguagem *case sensitive*, veja:

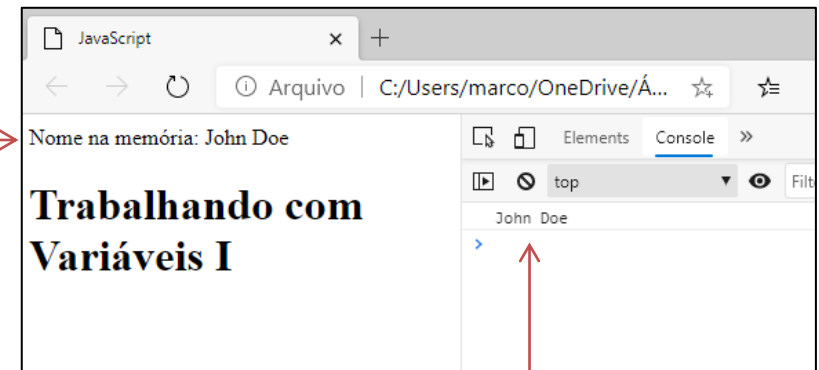
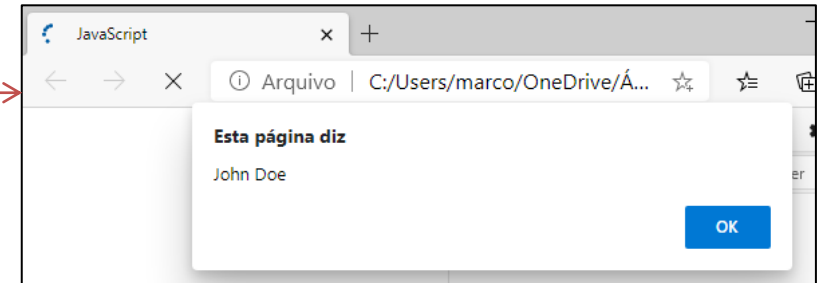
```
let nome é diferente de let NOME
```

# JavaScript: Variáveis

Praticando...

IMPORTANTE!!! A tipagem é definida pelo valor

```
index.html
1 <DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>JavaScript</title>
6     <script>
7       //Strings: pode usar aspas simples ou duplas
8       let nome = 'John Doe'
9
10      //Numbers
11      let numeroInteiro = 100
12      let numeroQuebrado = 100.50
13
14      //Boolean
15      let verdadeiro = true
16      let falso = false
17
18      //Saídas em dialog
19      alert(nome)
20
21      //Saída com objeto que representa o DOM
22      document.write('Nome na memória: ' + nome)
23
24      //Saída para processo de Depuração (Debug)
25      console.log(nome)
26    </script>
27  </head>
28  <body>
29    <h1>Trabalhando com Variáveis I</h1>
30  </body>
31
32 </html>
```



## JavaScript: Concatenação e Entrada de dados

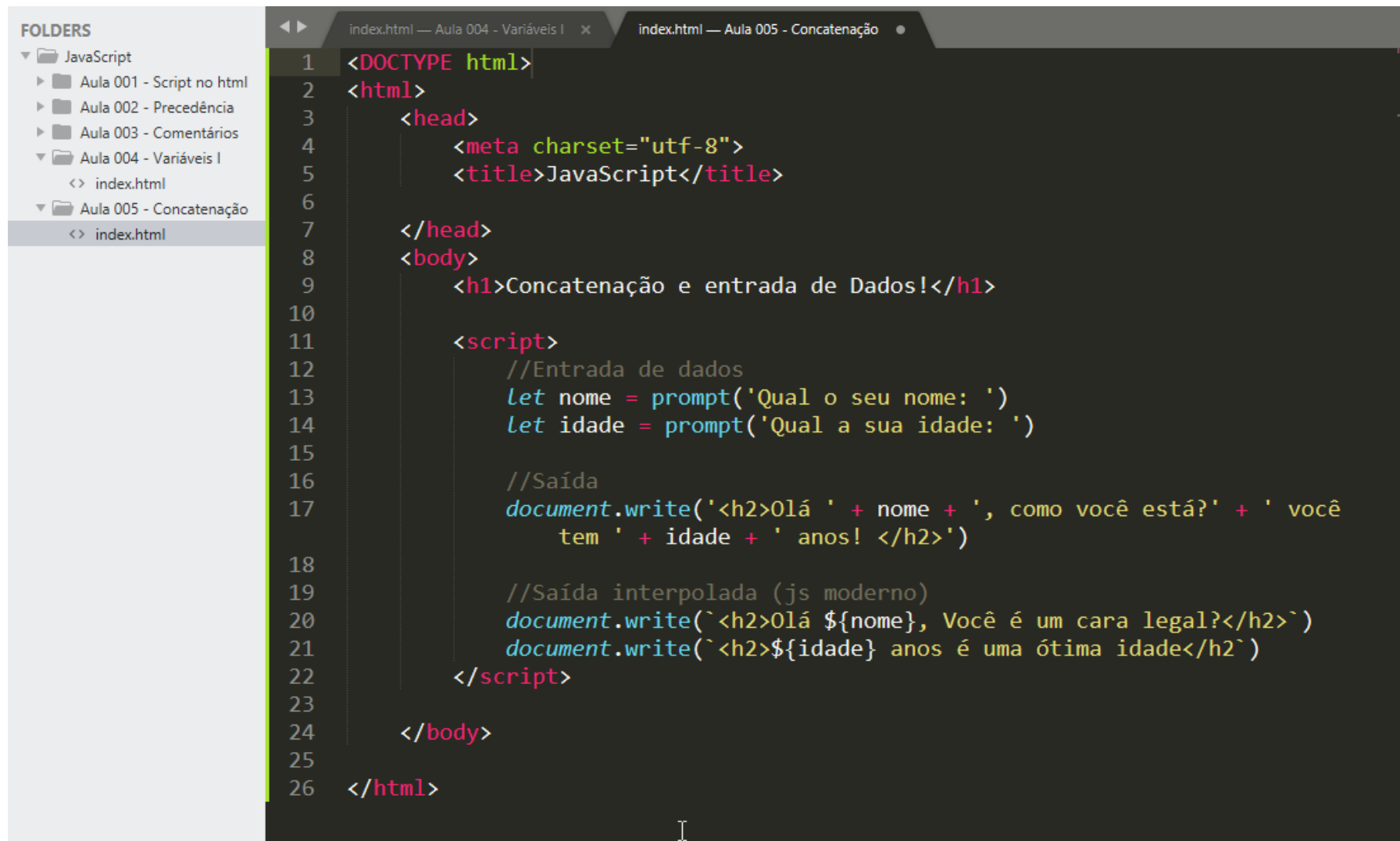
---

É possível concatenar (juntar) tipos diferentes e o JavaScript se encarregará de realizar a conversão entre os tipos, podendo resultar em algo não esperado. Para exemplificar melhor a concatenação, vamos fazer uso de entradas de dados através do comando *prompt*. O comando *prompt* vai sempre receber os valores como *strings* e por isso precisamos fazer *casting* de tipos para realizarmos cálculos. A diferença de tipos pode ser visualizada no console.

Saída de dados em *Template Strings*

*Template strings* são envolvidas por (acentos graves) (`` ``) em vez de aspas simples ou duplas. *Template strings* podem possuir *placeholders*. Estes são indicados por um cifrão seguido de chaves (`${expression}`). As expressões nos *placeholders*, bem como o texto em volta delas são passados a uma função. A função padrão apenas concatena as partes em uma string única.

# JavaScript: Concatenação e Entrada de dados

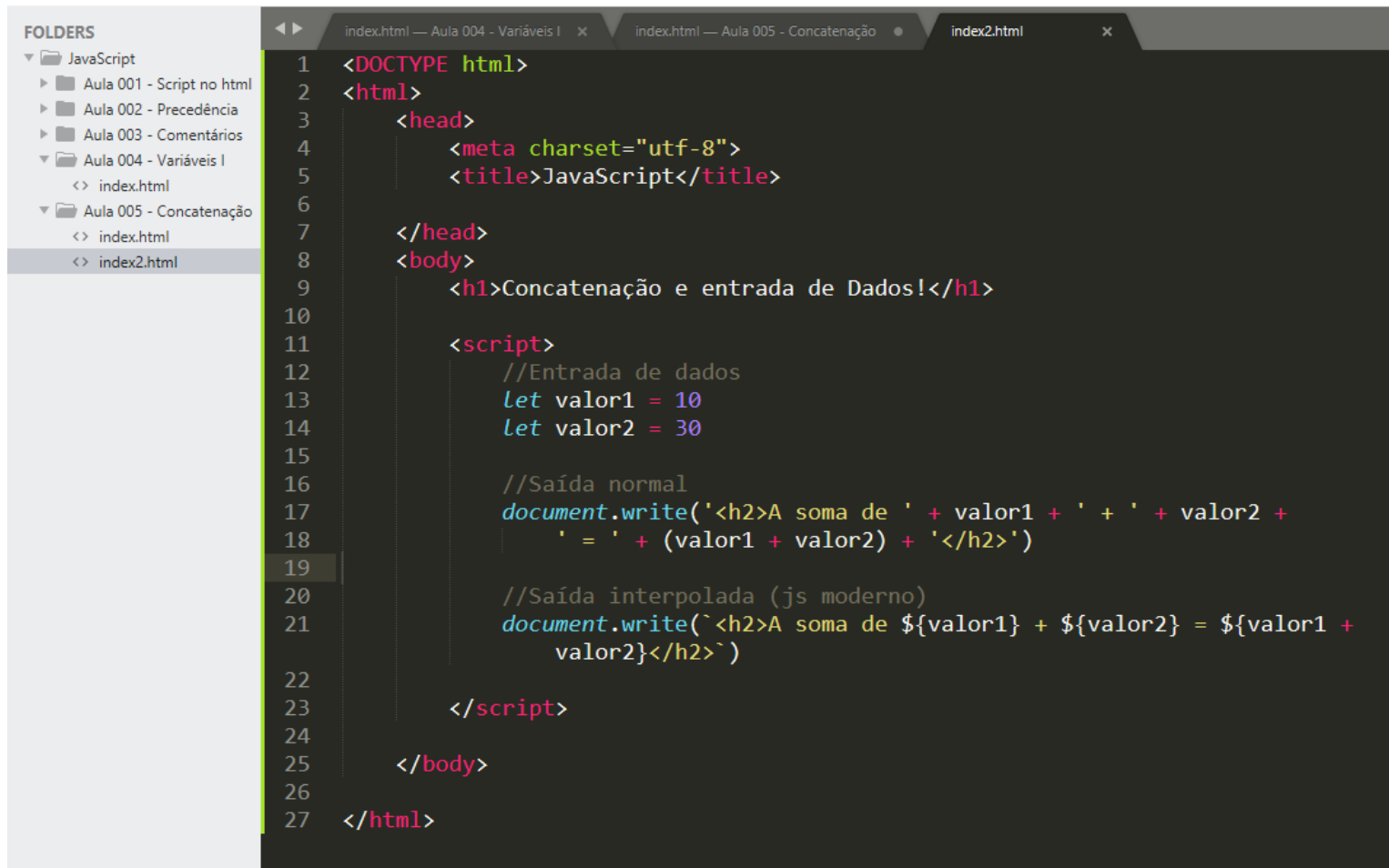


The image shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a folder structure for a JavaScript project, with the current file being 'index.html' in the 'Aula 005 - Concatenação' folder. The code editor displays the following HTML document:

```
1 <DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>JavaScript</title>
6
7   </head>
8   <body>
9     <h1>Concatenação e entrada de Dados!</h1>
10
11    <script>
12      //Entrada de dados
13      let nome = prompt('Qual o seu nome: ')
14      let idade = prompt('Qual a sua idade: ')
15
16      //Saída
17      document.write('<h2>Olá ' + nome + ', como você está?' + ' você
18        tem ' + idade + ' anos! </h2>')
19
20      //Saída interpolada (js moderno)
21      document.write('<h2>Olá ${nome}, Você é um cara legal?</h2>`)
22      document.write('<h2>${idade} anos é uma ótima idade</h2>`)
23    </script>
24  </body>
25
26 </html>
```

# JavaScript: Concatenação e Entrada de dados

Você pode também utilizar dados interpolados na saída, no exemplo abaixo vamos calcular uma soma de 2 variáveis na saída de dados.



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a folder named 'JavaScript' with subfolders 'Aula 001 - Script no html', 'Aula 002 - Precedência', 'Aula 003 - Comentários', 'Aula 004 - Variáveis I', and 'Aula 005 - Concatenação'. The code editor shows the content of 'index2.html' in the 'Aula 005 - Concatenação' folder. The code is as follows:

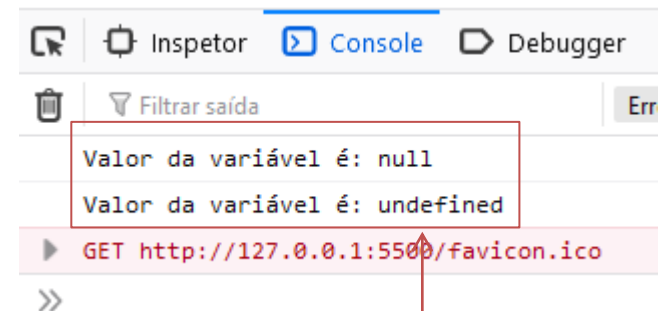
```
1 <DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>JavaScript</title>
6
7   </head>
8   <body>
9     <h1>Concatenação e entrada de Dados!</h1>
10
11    <script>
12      //Entrada de dados
13      let valor1 = 10
14      let valor2 = 30
15
16      //Saída normal
17      document.write('<h2>A soma de ' + valor1 + ' + ' + valor2 +
18        ' = ' + (valor1 + valor2) + '</h2>')
19
20      //Saída interpolada (js moderno)
21      document.write(`<h2>A soma de ${valor1} + ${valor2} = ${valor1 +
22        valor2}</h2>`)
23
24    </script>
25
26  </body>
27 </html>
```

# JavaScript: Null e Undefined

Null representa a ausência intencional de valor

Undefined, apesar de declarada não possui valor algum

```
25 <title>JavaScript</title>
26 </head>
27
28 <body>
29
30 <h1>Valores Null e Undefined</h1>
31 <hr>
32
33 <script>
34   //Valor Null
35   let valor1 = null
36
37   //Valor indefinido
38   let valor2
39
40   //Saída
41   document.write(`O valor da variável é: ${valor1} <br>`)
42   document.write(`O valor da variável é: ${valor2}`)
43
44   //Saída Console
45   console.log(`Valor da variável é: ${valor1}`)
46   console.log(`Valor da variável é: ${valor2}`)
47 </script>
```



É importante que os valores de Null e Undefined sejam tratados durante a lógica de programação

# JavaScript: Alterando valores de variáveis

Um determinado valor alocado na memória pode ter seu conteúdo alterado durante a execução do programa.

```
30 <h1>Alterando valores de variáveis</h1>
31 <hr>
32
33 <script>
34   //Valor Null
35   let valor = null
36
37   //Saída antes
38   console.log(`Antes: ${valor}`)
39
40   //Atribuindo um valor
41   valor = 'Valor alterado!'
42
43   //Saída Depois
44   console.log(`Depois: ${valor}`)
45
46   //Atribuindo outro valor
47   valor = 100.5
48
49   //Outra saída
50   console.log(`Outra alteração: ${valor}`)
51 </script>
```

## Alterando valores das variáveis



# JavaScript: Revisão

---

- ✓ Inclusão de códigos JavaScript em páginas Html
- ✓ Precedência de execução
- ✓ Comentários
- ✓ Variáveis
- ✓ Concatenação
- ✓ Valores Null e Undefined



# JavaScript: Desafio

Crie um programa que altere os valores das variáveis X e Y baseados nos seguintes dados:

**X = 50**  
**Y = 200**  
**Z = null**

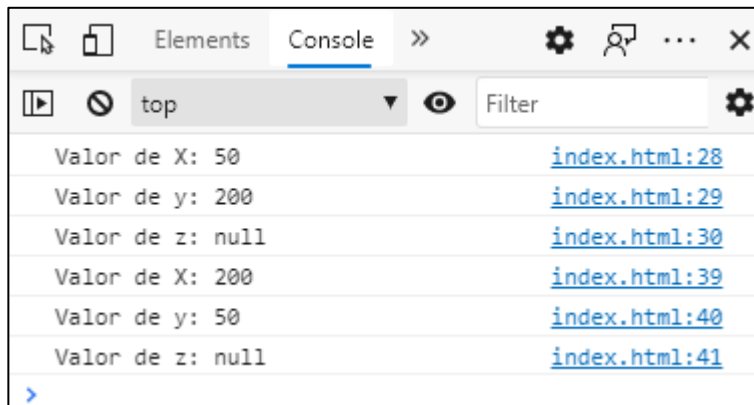
O desafio é inverter os valores de X e Y utilizando a variável Z como auxiliar, ou seja,

**NÃO PODE FAZER X = 200 E Y = 50, ÓBVIO!!!**

O programa deve mostrar o antes e o depois das variáveis no **BROWSER!**.

Resposta para e-mail do professor:

Assunto: <Seu nome>Desafio XYZ: Aluno tal...



## Desafio XYZ

### Valores Iniciais

Valor de x: 50  
Valor de y: 200  
Valor de z: null

### Valores Finais

Valor de x: 200  
Valor de y: 50  
Valor de z: null

# JavaScript: Estrutura de Controle IF ELSE

A estrutura condicional if/else é um recurso que indica quais instruções o sistema deve processar de acordo com uma expressão booleana. Assim, o sistema testa se uma condição é verdadeira e então executa comandos de acordo com esse resultado.

Há duas formas de trabalharmos com condicionais no JavaScript

```
19  <h1>Estudo Condicional: If... else...</h1>
20  <hr>
21
22  <script>
23
24      //Comando IF ELSE Simples
25      if (condição) {
26          //Executa verdadeiro
27      } else {
28          //Executa falso
29      }
30
31      //Comando IF ELSE Encadeado
32      if (condição) {
33          //Executa verdadeiro
34      } else if (condição) {
35          //Executa verdadeiro
36      } else {
37          //Executa caso as anteriores sejam falsas
38      }
39
40  </script>
```

Simples

Entenda: um ou outro

Encadeada

Entenda: várias condições

# JavaScript: Estrutura de Controle Condicional IF ELSE

Exemplificando a condicional: Exemplo para encontra o maior entre dois números, ainda sem entrada do usuário. As variáveis são pré-definidas para executar nosso teste.

```
22 <script>
23   //declaração
24   let info = true
25
26   //Condicional
27   if (info) {
28     document.write(`<h1 class="text-success p-2">Informação Verdadeira! </h1>`)
29   } else {
30     document.write(`<h1 class="text-warning p-2">Informação Falsa! </h1>`)
31   }
32 </script>
33
34 <h1 class="display-3 p-2 text-primary">Estudo If Else: Encadeado</h1>
35 <hr>
36 <script>
37   //declaração
38   let a = 40
39   let b = 40
40
41   //Condicional A é maior que B
42   if (a > b) {
43     document.write(`<h1 class="text-success p-2">Informação Verdadeira! </h1>`)
44     document.write(`<h1 class="text-success p-2"> ${a} é maior que ${b} </h1>`)
45   //Condicional B é maior que A
46   } else if ( b > a) {
47     document.write(`<h1 class="text-success p-2">Informação Verdadeira! </h1>`)
48     document.write(`<h1 class="text-success p-2"> ${b} é maior que ${a} </h1>`)
49
50   //Nenhuma das condições acima é verdadeira
51   } else {
52     document.write(`<h1 class="text-warning p-2"> Condições falsas!</h1>`)
53     document.write(`<h1 class="text-success p-2"> ${a} é igual ${b} </h1>`)
54   }
55 </script>
```

Estudo If Else: Simples

Informação Verdadeira!

Estudo If Else: Encadeado

Condições falsas!

40 é igual 40

# JavaScript: Estrutura Condicional IF ELSE

---

## Operadores de Comparação

**Um operador de comparação retorna um valor Booleano (Boolean)** indicando que a comparação é verdadeira (true) ou falsa (false). O JavaScript possui comparações estritas e conversão de tipos. Uma comparação estrita (===) somente é verdade se os operandos forem do mesmo tipo e de conteúdo correspondente. A comparação abstrata mais comumente utilizada (==) converte os operandos no mesmo tipo antes da comparação. Para comparações abstratas relacionais (<=), os operandos são primeiro convertidos em primitivos, depois para o mesmo tipo, depois comparados.

Tipos:

- ✓ Igualdade: == (iguais)
- ✓ Idêntico: === (iguais e de mesmo tipo)
- ✓ Diferente: != (Diferente)
- ✓ Não idêntico: !== (diferentes e de tipos diferentes)
- ✓ Menor: < (menor)
- ✓ Maior: > (maior)
- ✓ Menor igual: <= (menor ou igual)
- ✓ Maior igual: >= (maior ou igual)

# JavaScript: Estrutura de Controle Condicional IF ELSE

Operadores de Comparação praticando...

✓ Igualdade (==)

```
18 <div class="container">
19
20 <h1 class="display-5 p-2">Estruturas Condicionais II</h1>
21 <hr>
22 <h1 class="display-6 text-primary p-2">Prática com Operadores de Comparação</h1>
23
24 <script>
25     let a = 20
26     let b = 20
27
28     //Impressão
29     document.write('Valor de A: ' + a)
30     document.write('<br>')
31     document.write('Valor de B: ' + b)
32     document.write('<hr>')
33
34     //Condicional
35     if (a == b) {
36         document.write(a + ' é igual a ' + b + '!')
37     } else {
38         document.write(a + ' não é igual a ' + b + '!')
39     }
40     document.write('<hr>')
41 </script>
42
43 </div>
```

Estruturas Condicionais II

Prática com Operadores de Comparação

Valor de A: 20

Valor de B: 20

20 é igual a 20!

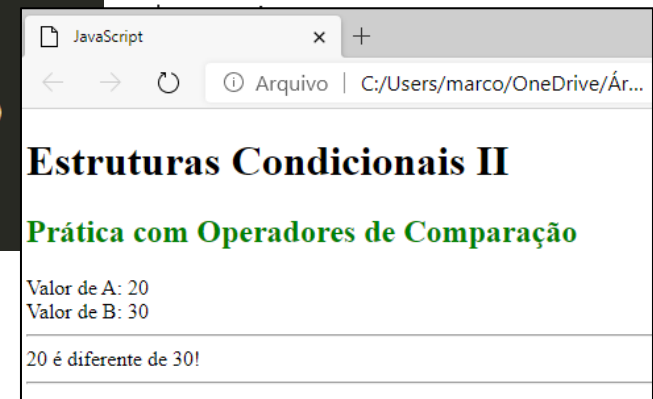
# JavaScript: Estrutura de Controle IF ELSE

Operadores de Comparação praticando...

✓ Diferente (!=)

```
8      <body>
9      <h1>Estruturas Condicionais II</h1>
10     <h2 style="color: green">Prática com Operadores de Comparação</h2>
11
12     <script>
13         let a = 20
14         let b = 30
15
16         //Impressão
17         document.write('Valor de A: ' + a)
18         document.write('<br>')
19         document.write('Valor de B: ' + b)
20         document.write('<br>')
21         document.write('<hr>')
22         //Condicional
23         if (a != b) {
24             document.write(a + ' é diferente de ' + b + '!')
25         } else {
26             document.write(a + ' não é diferente de ' + b + '!')
27         }
28         document.write('<hr>')
29     </script>
30 </body>
```

Saída no



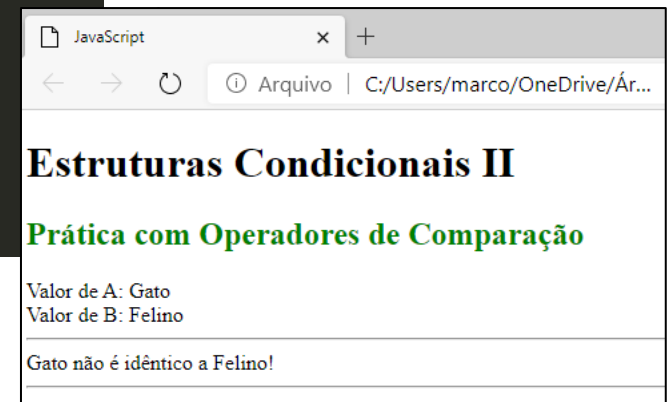
# JavaScript: Estrutura de Controle IF ELSE

Operadores de Comparação praticando...

✓ Idêntico (===)

```
8  <body>
9  <h1>Estruturas Condicionais II</h1>
10 <h2 style="color: green">Prática com Operadores de Comparação</h2>
11
12 <script>
13   let a = 'Gato'
14   let b = 'Felino'
15
16   //Impressão
17   document.write('Valor de A: ' + a)
18   document.write('<br>')
19   document.write('Valor de B: ' + b)
20   document.write('<br>')
21   document.write('<hr>')
22   //Condicional
23   if (a === b) {
24     document.write(a + ' é idêntico a ' + b + '!!')
25   } else {
26     document.write(a + ' não é idêntico a ' + b + '!!')
27   }
28   document.write('<hr>')
29 </script>
30 </body>
```

Saída no



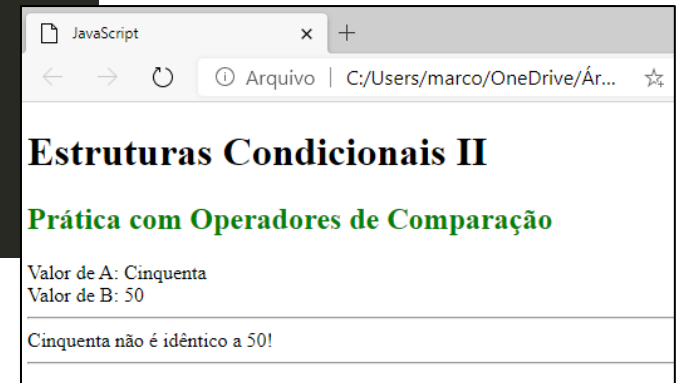
# JavaScript: Estrutura de Controle IF ELSE

Operadores de Comparação praticando...

✓ Não Idêntico (!==)

```
8      <body>
9          <h1>Estruturas Condicionais II</h1>
10         <h2 style="color: green">Prática com Operadores de Comparação</h2>
11
12         <script>
13             let a = 'Cinquenta'
14             let b = 50
15
16             //Impressão
17             document.write('Valor de A: ' + a)
18             document.write('<br>')
19             document.write('Valor de B: ' + b)
20             document.write('<br>')
21             document.write('<hr>')
22             //Condicional
23             if (a !== b) {
24                 document.write(a + ' não é idêntico a ' + b + '!')
25             } else {
26                 document.write(a + ' é idêntico a ' + b + '!')
27             }
28             document.write('<hr>')
29         </script>
30     </body>
```

Saída no





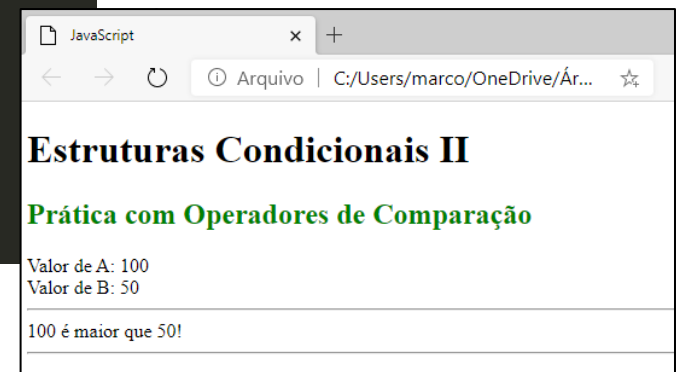
# JavaScript: Estrutura de Controle IF ELSE

Operadores de Comparação praticando...

✓ Maior que (>)

```
8      <body>
9      <h1>Estruturas Condicionais II</h1>
10     <h2 style="color: green">Prática com Operadores de Comparação</h2>
11
12     <script>
13         let a = 100
14         let b = 50
15
16         //Impressão
17         document.write('Valor de A: ' + a)
18         document.write('<br>')
19         document.write('Valor de B: ' + b)
20         document.write('<br>')
21         document.write('<hr>')
22         //Condicional
23         if (a > b) {
24             document.write(a + ' é maior que ' + b + '!')
25         } else {
26             document.write(a + ' não é maior que ' + b + '!')
27         }
28         document.write('<hr>')
29     </script>
30 </body>
```

Saída no



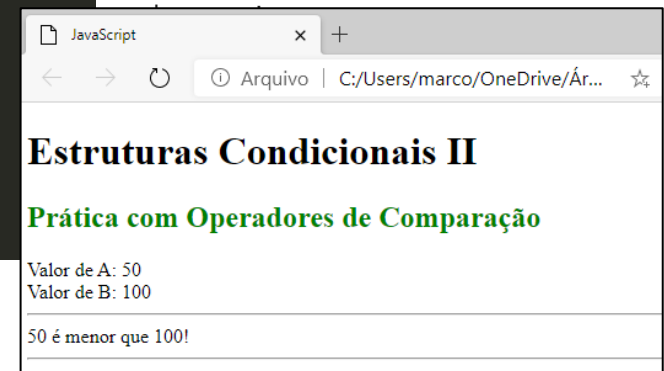
# JavaScript: Estrutura de Controle IF ELSE

Operadores de Comparação praticando...

✓ Menor que (<)

```
8  <body>
9    <h1>Estruturas Condicionais II</h1>
10   <h2 style="color: green">Prática com Operadores de Comparação</h2>
11
12   <script>
13     let a = 50
14     let b = 100
15
16     //Impressão
17     document.write('Valor de A: ' + a)
18     document.write('<br>')
19     document.write('Valor de B: ' + b)
20     document.write('<br>')
21     document.write('<hr>')
22     //Condicional
23     if (a < b) {
24       document.write(a + ' é menor que ' + b + '!')
25     } else {
26       document.write(a + ' não é menor que ' + b + '!')
27     }
28     document.write('<hr>')
29   </script>
30 </body>
```

Saída no



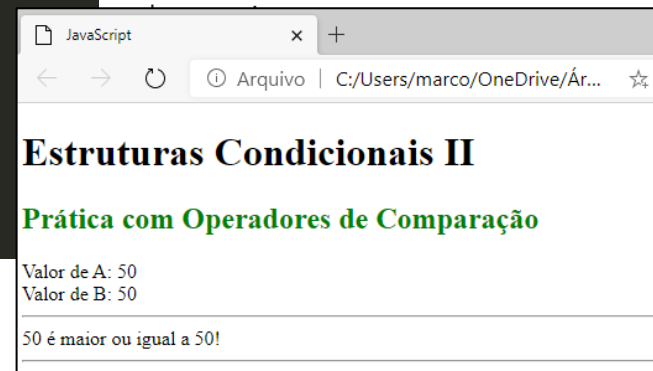
# JavaScript: Estrutura de Controle IF ELSE

Operadores de Comparação praticando...

✓ Maior ou igual (>=)

```
8  <body>
9    <h1>Estruturas Condicionais II</h1>
10   <h2 style="color: green">Prática com Operadores de Comparação</h2>
11
12   <script>
13     let a = 50
14     let b = 50
15
16     //Impressão
17     document.write('Valor de A: ' + a)
18     document.write('<br>')
19     document.write('Valor de B: ' + b)
20     document.write('<br>')
21     document.write('<hr>')
22     //Condicional
23     if (a >= b) {
24       document.write(a + ' é maior ou igual a ' + b + '!')
25     } else {
26       document.write(a + ' é menor que ' + b + '!')
27     }
28     document.write('<hr>')
29   </script>
30 </body>
```

Saída no



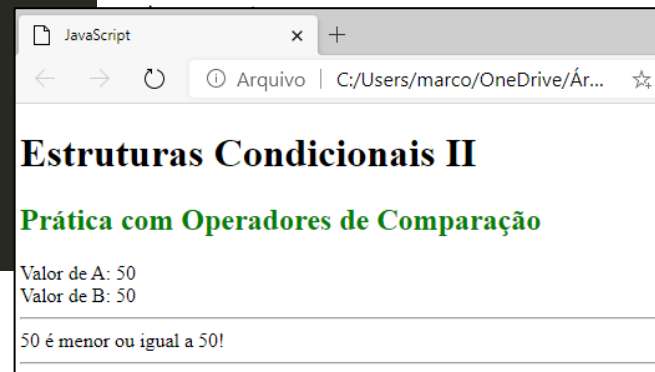
# JavaScript: Estrutura de Controle IF ELSE

Operadores de Comparação praticando...

✓ Menor ou igual (<=)

```
8  <body>
9    <h1>Estruturas Condicionais II</h1>
10   <h2 style="color: green">Prática com Operadores de Comparação</h2>
11
12   <script>
13     let a = 50
14     let b = 50
15
16     //Impressão
17     document.write('Valor de A: ' + a)
18     document.write('<br>')
19     document.write('Valor de B: ' + b)
20     document.write('<br>')
21     document.write('<hr>')
22     //Condicional
23     if (a <= b) {
24       document.write(a + ' é menor ou igual a ' + b + '!')
25     } else {
26       document.write(a + ' é maior que ' + b + '!')
27     }
28     document.write('<hr>')
29   </script>
30 </body>
```

Saída no



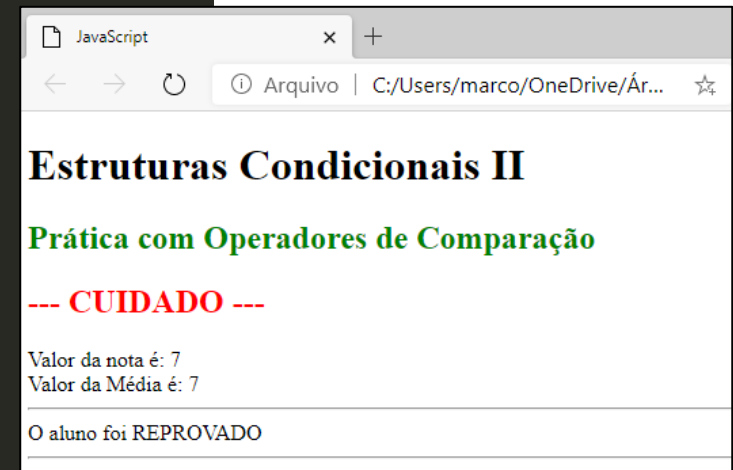
# JavaScript: Estrutura de Controle IF ELSE

## Operadores de Comparação praticando...

✓ ATENÇÃO!!!

```
8 <body>
9 <h1>Estruturas Condicionais II</h1>
10 <h2 style="color: green">Prática com Operadores de Comparação</h2>
11 <h2 style="color: red">--- CUIDADO ---</h2>
12
13 <script>
14   let nota = prompt('Informe uma nota: ')
15   let media = 7
16
17   //Impressão
18   document.write('Valor da nota é: ' + nota)
19   document.write('<br>')
20   document.write('Valor da Média é: ' + media)
21   document.write('<br>')
22   document.write('<hr>')
23
24   //Condicional
25   if (nota === media) {
26     document.write('Aluno foi APROVADO!')
27   } else {
28     document.write('O aluno foi REPROVADO')
29   }
30   document.write('<hr>')
31 </script>
32 </body>
```

→ Esse valor do prompt é uma **String**, como explicado anteriormente!



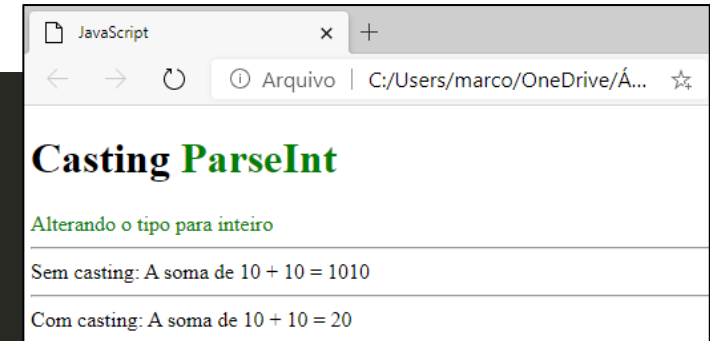
→ Comparação equivocada! Mas funcionaria com operadores simples (=, >, <...)

# JavaScript: Casting – Conversão de tipos de dados

Esse recurso é utilizado quando se quer alterar o tipo de dado de uma variável. Vamos pensar no comando prompt que sempre irá retornar uma string. Então, se precisássemos fazer uma conta com uma entrada numérica do usuário, precisaríamos fazer o casting dessa variável.

## *parseInt(): Conversão para inteiro*

```
8   <body>
9   <h1>Casting <span style="color: green">ParseInt</span></h1>
10  <span style="color: green">Alterando o tipo para inteiro</span>
11  <hr>
12  <script>
13      //Sem o Casting
14      let valor1 = prompt('Insira um número: ')
15      let valor2 = prompt('Insira outro valor: ')
16
17      //Realizando o cálculo
18      resultado = valor1 + valor2
19
20      //Saída
21      document.write(`Sem casting: A soma de ${valor1} + ${valor2} = ${
22          resultado}`)
23
24      document.write('<hr>')
25
26      //Com o Casting
27      valor1 = parseInt(valor1)
28      valor2 = parseInt(valor2)
29
30      //Realizando o cálculo
31      resultado = valor1 + valor2
32
33      //Saída
34      document.write(`Com casting: A soma de ${valor1} + ${valor2} = ${
35          resultado}`)
36  </script>
37 </body>
```

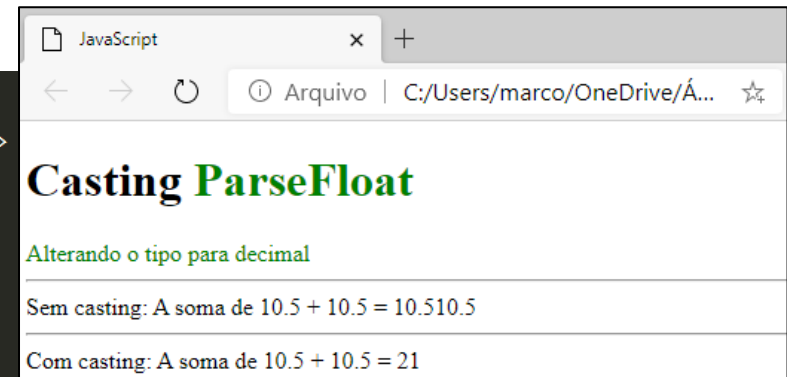


# JavaScript: Casting – Conversão de tipos de dados

Esse recurso é utilizado quando se quer alterar o tipo de dado de uma variável. Vamos pensar no comando prompt que sempre irá retornar uma string. Então, se precisássemos fazer uma conta com uma entrada numérica do usuário, precisaríamos fazer o casting dessa variável.

## *parseFloat(): Conversão para decimal*

```
8 <body>
9 <h1>Casting <span style="color: green">parseFloat</span></h1>
10 <span style="color: green">Alterando o tipo para decimal</span>
11 <hr>
12
13 <script>
14 //Sem o Casting
15 let valor1 = prompt('Insira um número decimal: ')
16 let valor2 = prompt('Insira outro número decimal: ')
17
18 //Realizando o cálculo
19 resultado = valor1 + valor2
20
21 //Saída
22 document.write(`Sem casting: A soma de ${valor1} + ${valor2} = ${
23     resultado}`)
24
25 document.write('<hr>')
26
27 //Com o Casting
28 valor1 = parseFloat(valor1)
29 valor2 = parseFloat(valor2)
30
31 //Realizando o cálculo
32 resultado = valor1 + valor2
33
34 //Saída
35 document.write(`Com casting: A soma de ${valor1} + ${valor2} = ${
36     resultado}`)
37 </script>
</body>
```

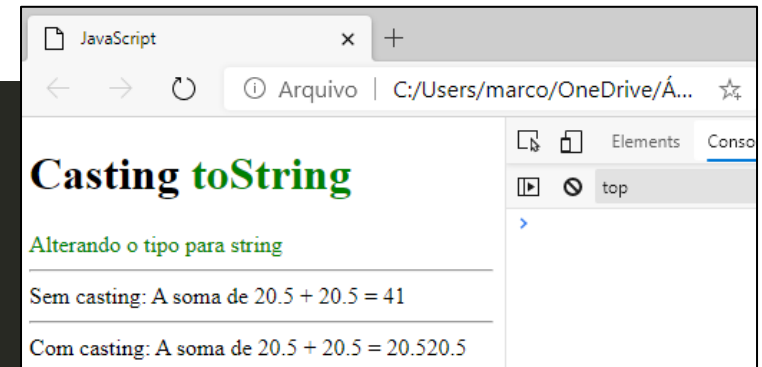


# JavaScript: Casting – Conversão de tipos de dados

Também podemos fazer o contrário do que vimos, ou seja, fazer a representação textual da soma de valores. Nesse caso estaremos realizando uma operação de concatenação.

## *toString(): Conversão para string*

```
8   <body>
9   <h1>Casting <span style="color: green">toString</span></h1>
10  <span style="color: green">Alterando o tipo para string</span>
11  <hr>
12
13  <script>
14      //Sem o Casting
15      let valor1 = 20.5
16      let valor2 = 20.5
17
18      //Realizando o cálculo
19      resultado = valor1 + valor2
20
21      //Saída
22      document.write(`Sem casting: A soma de ${valor1} + ${valor2} =
23                      ${resultado}`)
24
25      document.write('<hr>')
26
27      //Com o Casting
28      resultado = valor1.toString() + valor2.toString()
29
30      //Saída
31      document.write(`Com casting: A soma de ${valor1} + ${valor2} =
32                      ${resultado}`)
33  </script>
34 </body>
```





# JavaScript: Operadores lógicos

---

Os operadores lógicos são utilizados para nos ajudar na realização de comparações condicionais. Aumentando o nível de sua comparação lógica, também aumenta a complexidade no entendimento dessas operações. **É preciso praticar bastante para dominar técnicas mais avançadas de comparações.**

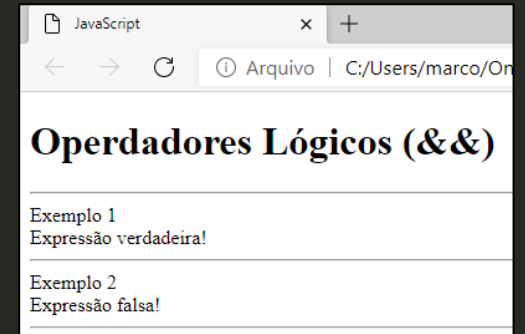
Os operadores lógicos são:

- ✓ E ( && ) – Verdadeiro se todas as expressões forem verdadeiras
- ✓ OU ( || ) – Verdadeiro se pelo menos uma das expressões for verdadeira
- ✓ Negação ( ! ) – Inverte o resultado da expressão

# JavaScript: Operadores lógicos

## Testando o && verdadeiro e Falso

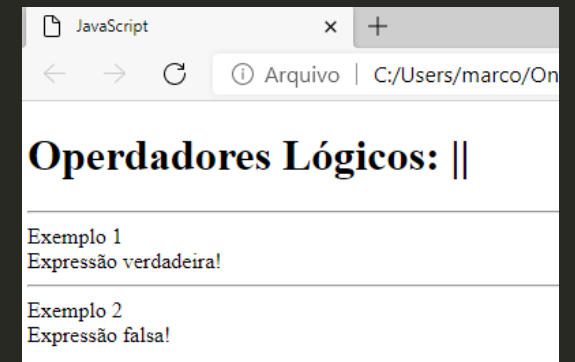
```
8      <body>
9          <h1>Operadores Lógicos (&&)</h1>
10         <hr>
11
12         <script>
13             //Operador &&: Verdadeiro
14             document.write("Exemplo 1 <br>")
15             if (10 == 10 && 20 < 40 && 'a' != 'b') {
16                 document.write("Expressão verdadeira!")
17             } else {
18                 document.write("Expressão falsa!")
19             }
20             document.write('<hr>')
21
22             //Operador &&: Falso
23             document.write("Exemplo 2 <br>")
24             if (10 == 10 && 20 < 40 && 'a' == 'b') {
25                 document.write("Expressão verdadeira!")
26             } else {
27                 document.write("Expressão falsa!")
28             }
29             document.write('<hr>')
30
31         </script>
32     </body>
```



# JavaScript: Operadores lógicos

## Testando o || verdadeiro e Falso

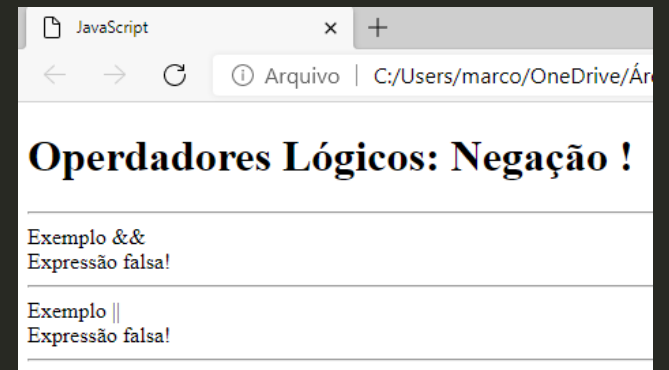
```
8   <body>
9     <h1>Operadores Lógicos: || </h1>
10    <hr>
11
12    <script>
13
14      //Operador || (OU): Verdadeiro
15      document.write("Exemplo 1 <br>")
16      if (10 == 10 || 20 < 40 || 'a' == 'b') {
17        document.write("Expressão verdadeira!")
18      } else {
19        document.write("Expressão falsa!")
20      }
21      document.write('<hr>')
22
23      //Operador || (OU): Falso
24      document.write("Exemplo 2 <br>")
25      if (10 == 9 || 20 > 40 || 'a' == 'b') {
26        document.write("Expressão verdadeira!")
27      } else {
28        document.write("Expressão falsa!")
29      }
30
31    </script>
32  </body>
```



# JavaScript: Operadores lógicos

## Testando a negação do && e do ||

```
8      <body>
9      <h1>Operadores Lógicos: Negação ! </h1>
10     <hr>
11
12     <script>
13         //Negar o Operador &&
14         document.write("Exemplo && <br>")
15         if (!(10 == 10 && 20 < 40 && 'a' != 'b')) {
16             document.write("Expressão verdadeira!")
17         } else {
18             document.write("Expressão falsa!")
19         }
20         document.write('<hr>')
21
22         //Negar o Operador || (OU)
23         document.write("Exemplo || <br>")
24         if (!(10 == 10 || 20 < 40 || 'a' == 'b')) {
25             document.write("Expressão verdadeira!")
26         } else {
27             document.write("Expressão falsa!")
28         }
29         document.write('<hr>')
30
31     </script>
32 </body>
```

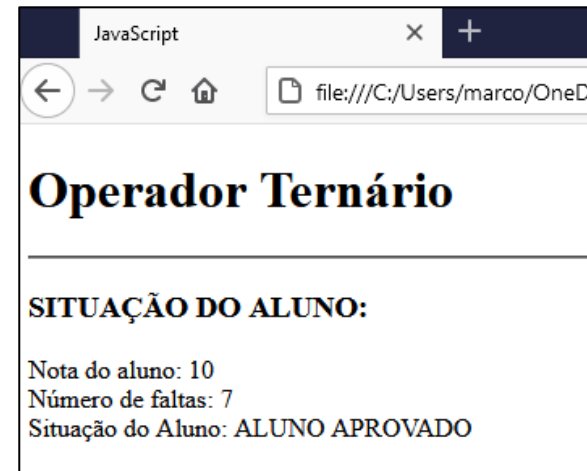


# JavaScript: Operador Ternário

O operador ternário é uma estrutura muito parecida com o a estrutura IF ELSE com limitações. Sua diferença está simplificação da codificação, tornando sua sintaxe mais simples. Usar operador ternário reduz o número de linhas de código, mas é preciso uma compreensão maior de quem for dar manutenção no sistema.

Sintaxe: `let variável = <condição> ? <verdadeiro> : <falso>`

```
8      <body>
9      <h1>Operador Ternário</h1>
10     <hr>
11
12     <script>
13         //Situação de um aluno
14         //Entradas
15         let notaAluno = prompt('Entre com a nota: ')
16         let numeroFaltas = prompt('Entre com nº de faltas: ')
17         let media = 7
18         let limiteFaltas = 10
19
20         //Cabeçalho
21         document.write('<h3>SITUAÇÃO DO ALUNO:</h3>')
22
23         //Operador ternário
24         let situacao =
25         (notaAluno >= media && numeroFaltas <= limiteFaltas) ?
26         'ALUNO APROVADO' : 'ALUNO REPROVADO'
27
28         //Saída
29         document.write(`Nota do aluno: ${notaAluno}`)
30         document.write('<br>')
31         document.write(`Número de faltas: ${numeroFaltas}`)
32         document.write('<br>')
33         document.write('Situação do Aluno: ')
34         document.write(situacao)
35
36     </script>
37
38 </body>
```



# JavaScript – Operadores aritméticos

---

São operadores matemáticos utilizados para efetuar cálculos dentro da lógica da programação. Eles se dividem em:

Operador de **Adição (+)**: Soma de Valores e concatenação (valores string)

Operador de **Subtração (-)**: Diferença entre valores

Operador de **Multiplificação** ou **Produto (\*)**: Produto entre valores

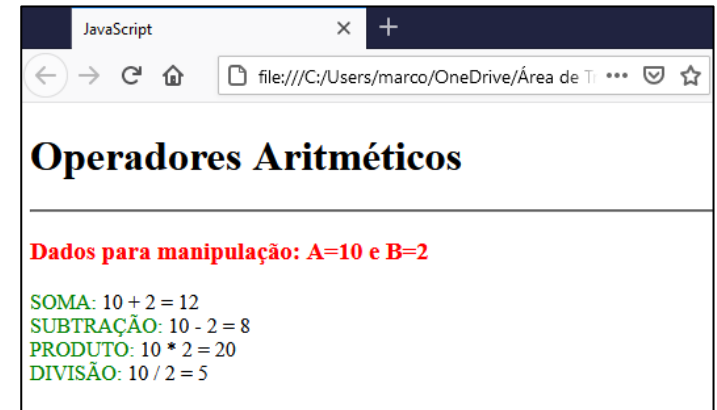
Operador de **Divisão (/)**: Quociente entre valores

Operador de **Módulo (%)**: Retorna o resto de uma divisão

# JavaScript – Operadores aritméticos

Exemplo prático: Soma – Subtração – Produto – Divisão

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>JavaScript</title>
5     <meta charset="utf-8">
6     <link rel="stylesheet" type="text/css" href="estilo.css">
7   </head>
8
9   <body>
10    <h1>Operadores Aritméticos</h1>
11    <hr>
12    <script>
13      //Declaração
14      let A = 10
15      let B = 2
16
17      document.write('<h3><span class="vermelha">')
18      document.write('Dados para manipulação: A=10 e B=2')
19      document.write('</span></h3>')
20
21      document.write('<span class="verde"> SOMA: </span>')
22      document.write(`${A} + ${B} = ${A + B} <br>`)
23
24      document.write('<span class="verde"> SUBTRAÇÃO: </span>')
25      document.write(`${A} - ${B} = ${A - B} <br>`)
26
27      document.write('<span class="verde"> PRODUTO: </span>')
28      document.write(`${A} * ${B} = ${A * B} <br>`)
29
30      document.write('<span class="verde"> DIVISÃO: </span>')
31      document.write(`${A} / ${B} = ${A / B} <br>`)
```



O que acontece se utilizarmos o *prompt* para entrada de dados, sabendo que seu retorno é uma *string*?

# JavaScript – Operadores aritméticos

**Ordem de Precedência:** forma com que as operações serão realizadas.

Qual seria o resultado dessas operações

$((8 + 4 - 2) * (4 + 2)) / 2$        $(5 + 5 + 5) + (5 / 4)$

$(8 + 4 - 2) * (4 + 2) / 2$

$(8 + 4 - 2 * 4 + 2 / 2)$





# JavaScript: Atividade 6

---

1. Faça uma aplicação que peça um número inteiro. Depois verifique se esse número é par ou se ele é ímpar.
2. Faça uma aplicação que peça ao usuário 3 números. Depois, mostre na tela qual é o maior e o menor número.
3. Faça um programa que peça a velocidade média de um carro. Se o carro ultrapassar o limite de 60Km por hora, mostre na tela a mensagem: 'Limite de velocidade acima do permitido'. Se a velocidade estiver a abaixo dos 60Km, mostrar a frase: 'Velocidade normal!'
4. Faça uma aplicação que calcule um aumento salarial de um funcionário. Se o salário for até R\$1000, efetuar um aumento de 10%. Se o salário estiver acima de R\$1000 e abaixo ou igual a R\$1500, efetuar um aumento de 5%. Se o salário estiver acima de R\$1500, não recebe aumento.
5. Faça uma aplicação que peça para o usuário a distância de uma viagem de Juiz de Fora a São Paulo. Calcule o preço da passagem sabendo que a empresa de ônibus cobra R\$0,70 por km, em viagens de até 200Km. Acima dessa distância às viagens passam a custar R\$0,40.

## Os arquivos deverão ser enviados para:

Assunto: <seu nome \_ sobrenome:>

Exemplo: **Sebastião Marcos: Atividades condicionais**

**Observação:** Não esqueça de anexar os arquivos

**Sugestão para os nomes dos arquivos:**

numero\_par\_impar.html — maior\_numero.html — velocidade\_media.html

aumento\_salarial.html — viagem.html

----- Envios fora desse padrão serão desconsiderados

# JavaScript: Atividades

Faça um aplicação para calcular a situação de uma aluno em uma escola.  
O sistema receberá via teclado o nome, o número de faltas do aluno e quatro notas.  
Com base nesses dados verifique a situação desse aluno da seguinte forma:

Se a nota do aluno estiver entre 9 - 10 e o número de faltas for menor ou igual 3:

Situação: Aprovado com mérito

Observação: Tudo certo com esse aluno

Se a nota do aluno estiver entre 7 - 9:

Situação: Aluno Aprovado!

Observação: Tudo certo com esse aluno

Se a nota do aluno estiver entre 5 - 7:

Situação: Aluno em recuperação!

Observação: Enviar email para o(s) responsável(eis)!

Se a nota do aluno for < 5:

Situação: Aluno Reprovado!

Observação: Enviar email para o(s) responsável(eis)!

Siga a tela ao lado para o layout

Importante:

- não usar código externo ou comandos ainda não estudados
- Usar formatação com Bootstrap
- Enviar atividade 6 por email

Saída

## Escola Municipal Juiz de Fora

### Cálculo da Média Escolar

Nome do aluno: John Doe

Número de Faltas: 2

Notas: 9 : 9 : 9 : 9

Média do Aluno: 9

Situação: Aprovado com Mérito!

Observação: Tudo certo com esse aluno!

# JavaScript: Atividades

---

Validações:

- Nome em branco
- Numero de faltas negativo ou em branco
- nota < 0 e maior 10

Saída

Escola Municipal Juiz de Fora

Cálculo da Média Escolar

Nome do aluno: John Doe

Número de Faltas: 2

Notas: 9 : 9 : 9 : 9

Média do Aluno: 9

Situação: Aprovado com Mérito!

Observação: Tudo certo com esse aluno!

# JavaScript - Switch

A instrução do `switch` é usada para executar diferentes ações com base em diferentes condições. A condicional `switch` avalia uma expressão, combinando o valor da expressão para um cláusula `case`, e executa as instruções associadas ao `case`. Esse comando é muito semelhante ao `if`, mas só podemos fazer comparações idênticas (`===`). Para interromper a execução do bloco de código é utilizado o comando `break` e caso nenhuma opção `case` seja atendida, pode-se utilizar o comando `default` que tem o comportamento igual ao `else`, do comando `if`. Exemplo prático, um menu de opções!

```
17 <!--Abre Container-->
18 <div class="container">
19   <h1 class="display-5 p-2"> Trabalhando com tabelas</h1>
20   <hr>
21
22   <script>
23     let opcao = 1
24
25     switch (opcao) {
26       case 1:
27         //Bloco de Código
28         break
29       case 2:
30         //Bloco de Código
31         break
32       default:
33         //Bloco de Código
34         break
35     }
36   </script>
37
38   <script>
39     let opcao = 'a'
40
41     switch (opcao) {
42       case 'a':
43         //Bloco de Código
44         break
45       case 'b':
46         //Bloco de Código
47         break
48       default:
49         //Bloco de Código
50         break
51     }
52   </script>
53
54 </div>
55 <!--Fecha container-->
```

# JavaScript - Switch

Exemplo switch normal:

```
17 <!--Abre Container-->
18 <div class="container">
19   <h1 class="display-5 p-2">Switch Normal</h1>
20   <hr>
21
22   <section>
23     <h1 class="display-5 text-danger mt-4">Escolha uma opção</h1>
24     <ol class="list-group list-group-numbered">
25       <li class="list-group-item">Cor verde</li>
26       <li class="list-group-item">Cor Azul</li>
27       <li class="list-group-item">Cor Amarela</li>
28       <li class="list-group-item">Sair</li>
29     </ol>
30   </section>
31
32   <script>
33     let opcao = prompt('Escolha uma cor de 1 a 4: ')
34
35     switch (parseInt(opcao)) {
36       case 1:
37         document.write(`<p class="display-6 text-success">Você escolheu a cor VERDE</p>`)
38         break
39
40       case 2:
41         document.write(`<p class="display-6 text-primary">Você escolheu a cor AZUL</p>`)
42         break
43
44       case 3:
45         document.write(`<p class="display-6 text-danger">Você escolheu a cor VERMELHA</p>`)
46         break
47
48       default:
49         document.write(`<p class="display-6 text-danger">FIM DO PROGRAMA!</p>`)
50     }
51   </script>
52
53 </div>
54
55 <!--Fecha container-->
```

## Switch Normal

### Escolha uma opção

- |                |
|----------------|
| 1. Cor verde   |
| 2. Cor Azul    |
| 3. Cor Amarela |
| 4. Sair        |

Você escolheu a cor VERDE

O que acontece se não utilizarmos as instruções *break* ou *default*?

# JavaScript - Switch

Exemplo switch normal:

```
17 <!--Abre Container-->
18 <div class="container">
19   <h1 class="display-5 p-2">Switch com if...else</h1>
20   <hr>
21
22   <section>
23     <h1 class="display-5 text-danger mt-4">Escolha uma opção</h1>
24     <ol class="list-group list-group-numbered">
25       <li class="list-group-item">Par ou Ímpar</li>
26       <li class="list-group-item">Maior número</li>
27       <li class="list-group-item">Sair</li>
28     </ol>
29   </section>
30
31   <script>
32     let opcao = prompt('Escolha uma opção (1-3)')
33
34     switch (parseInt(opcao)) {
35       case 1:
36         //Declarando a variável
37         let numero = parseFloat(prompt('Digite um número: '))
38
39         //Condicional
40         if (numero % 2 == 0) {
41           document.write(`<p class="display-6 text-primary">
42             O número ${numero} é par</p>`)
43         } else {
44           document.write(`<p class="display-6 text-primary">
45             O número ${numero} é ímpar</p>`)
46         }
47         break
48     }
```

```
49     case 2:
50       //Declarando a variável
51       let a = parseFloat(prompt('Digite um número: '))
52       let b = parseFloat(prompt('Digite outro número: '))
53
54       if (a > b) {
55         document.write(`<p class="display-6 text-primary">
56           O número ${a} é maior que o número ${b}</p>`)
57       } else if (b > a) {
58         document.write(`<p class="display-6 text-primary">
59           O número ${b} é maior que o número ${a}</p>`)
60       } else {
61         document.write(`<p class="display-6 text-primary">
62           Os números são iguais!</p>`)
63       }
64       break
65
66       default:
67         document.write(`<p class="display-6 text-primary">FIM DO PROGRAMA!</p>`)
68     }
69   </script>
70 </div>
71 <!--Fecha container-->
```

Switch com if...else

Escolha uma opção

- |                 |
|-----------------|
| 1. Par ou Ímpar |
| 2. Maior número |
| 3. Sair         |

O número 2 é par

# JavaScript – Funções

Funções são blocos de construção fundamentais em JavaScript. Uma função é um procedimento de JavaScript - um conjunto de instruções que executa uma tarefa ou calcula um valor. Para usar uma função, você deve defini-la em algum lugar no escopo do qual você quiser chamá-la. Uma função pode ou não receber parâmetros. Assim ela pode ou não retornar informações. Funções sem retorno são chamadas *void*.

Exemplo de uma função:

```
17 <!--Abre Container-->
18 <div class="container">
19   <h1 class="display-5 p-2"> Funções</h1>
20   <hr>
21
22   <script>
23
24     function cacularAreaRetangulo(base, altura) {
25       let area = base * altura
26
27       document.write(`<p class="lead">Base do retângulo ${base}</p>`)
28       document.write(`<p class="lead">Altura do Retângulo ${altura}</p>`)
29       document.write(`<hr>`)
30
31       return area
32     }
33
34     retangulo = cacularAreaRetangulo(5, 2)
35
36     document.write(`<p class="lead">Área do retângulo é ${retangulo}</p>`)
37
38   </script>
39
40 </div>
41 <!--Fecha container-->
```

→ Nome da Função em camelCase

→ Parâmetros

→ Bloco de Códigos

→ Argumentos

# JavaScript – Funções Void

```
22 <!--Abre Container-->
23 <div class="container">
24   <h1 class="display-5 p-2"> Funções: Void</h1>
25   <hr>
26
27   <script>
28     function exibirParImpar(numero) {
29       if (isNaN(numero)) {
30         document.write(`<p class="lead">\`${numero}\` Não é um número!`)
31       } else if (numero % 2 == 0) {
32         document.write(`<p class="lead">O número ${numero} é Par!`)
33       } else {
34         document.write(`<p class="lead">O número ${numero} é Ímpar!`)
35       }
36     }
37     exibirParImpar(2)
38   </script>
39
40   <hr>
41 </div>
42 <!--Fecha container-->
```

## Funções: Void

O número 2 é Par!

**isNaN()**: Função global de Js que verifica se é um NaN!



# JavaScript – Funções com retorno

```
18 <!--Abre Container-->
19 <div class="container">
20   <h1 class="display-5 p-2"> Funções com retorno</h1>
21   <hr>
22
23   <script>
24     function cacularAreaRetangulo(base, altura) {
25       let area = base * altura
26
27       //Retorno o resultado da operação
28       return area
29     }
30
31     //Atribuindo o valor a uma variável
32     resultado = cacularAreaRetangulo(2, 100)
33
34     //Sem atribuição a uma variável
35     document.write(`<p class="lead">O cálculo da área é ${resultado}m²</p>`)
36   </script>
37
38   <hr>
39 </div>
40 <!--Fecha container-->
```

Funções com retorno

O cálculo da área é 200m²

# JavaScript – Funções com entrada de dados (fins didáticos)

```
22 <!--Abre Container-->
23 <div class="container">
24   <h1 class="display-5 p-2"> Funções: Etrada de dados</h1>
25   <hr>
26
27   <script>
28     function cacularAreaRetangulo(base, altura) {
29       let area = base * altura
30
31       //Retorno o resultado da operação
32       return area
33     }
34
35     //Entrada dos Valores
36     let base = parseFloat(prompt('Entre com o valor da base: '))
37     let altura = parseFloat(prompt('Entre com o valor da altura: '))
38
39     //Atribuindo valores a variável resultado
40     let area = cacularAreaRetangulo(base, altura)
41
42     //interpolando o resultado na saída
43     document.write(`<p class="lead">O cálculo da área é ${area}m²</p>`)
44
45   </script>
46 </div>
47 <!--Fecha container-->
```

Funções: Etrada de dados

O cálculo da área é 60m<sup>2</sup>

## Atividade 7

---

Desenvolva um **Switch** com opções de programas que executam as seguintes funções:

- a. Calcular área de um círculo. [Renderizar no browser um esboço do círculo.](#)
- b. Calcular peso ideal de uma pessoa.
- c. Calcular raiz quadrada de um número(Sem função matemática).
- d. Calcular a potência de um número (Sem função matemática).
- e. Calcular a média ponderada de 5 números (Sem laço).
- f. Calcular a conversão de graus Fahrenheit para Célsius.
- g. Calcular o salário líquido de um trabalhador, baseando-se em regras trabalhistas.
- h. Calcular a área de um triângulo equilátero. [Renderizar no browser um esboço do triângulo.](#)
- i. Calcular o valor da hipotenusa de um triângulo. [Renderizar no browser um esboço do triângulo.](#)
- j. Fim

Observação:

- Enviar o arquivo para o e-mail do professor.
- Ao menos 3 formatações Bootstrap.
- Comentar seu código explicando algumas partes que julgar pertinente.
- **Não utilizem recursos que ainda não foram vistos no arquivo de enviado .**

Assunto: Atividade 7: <Nome do Aluno>

## Desafio (faz parte da atividade 7)

---

Faça um programa que execute uma função que retorna se determinado caractere é uma vogal ou uma consoante. Valide o argumento passado para os casos em que o usuário digite um número ou não digite nada.

Observação:

- Enviar o arquivo para o e-mail do professor.
- Ao menos 3 formatações Bootstrap.
- Comentar seu código explicando algumas partes que julgar pertinente.
- **Não utilizem recursos que ainda não foram vistos no arquivo de enviado .**

Assunto: Atividade 7: <Nome do Aluno>

Desafio: Função e switch

---

"a" é uma Vogal!

---

Desafio: Função e switch

---

1 não é uma letra!

---

Desafio: Função e switch

---

"b" é uma Consoante!

---

Desafio: Função e switch

---

undefined! Valor indefinido!

---

# JavaScript – Escopo de variáveis

Ao utilizarmos a linguagem de programação JavaScript é muito importante e imprescindível que entendamos como aplicar escopo de variável de forma correta, para que possamos evitar erros em nosso código, erros os quais acabam se tornando mais difíceis de serem identificados, pois não se tratam de erros de sintaxe do código e sim de coerência e coesão do mesmo, onde o código é interpretado corretamente, porém não estará funcionando de forma como deveria, de forma a fazer com que certas partes da aplicação ou site não funcionem corretamente ou até mesmo parem de funcionar, assim tornasse muito importante que tenhamos pleno conhecimento da utilização correta de escopo no JavaScript.

Quando nos referimos a escopo de variável estamos se referindo a qual local de nosso código uma determinada variável pode ser acessada. No JavaScript existem somente dois tipos de escopos, que são eles, escopo global e local. O código a seguir ilustra basicamente como funcionam esses dois escopos. ([leia mais](#))

```
9      <body>
10      <h1>Funções: com retorno e entrada de dados</h1>
11      <hr>
12
13      <script>
14          function cacularAreaRetangulo(base, altura) {
15              let area = base * altura
16
17              //Retorno o resultado da operação
18              return area
19          }
20
21          //Entrada dos Valores
22          let base = prompt('Entre com o valor da base: ')
23          let altura = prompt('Entre com o valor da altura: ')
24
25          //Convertendo a string do prompt em number
26          base = parseFloat(base)
27          altura = parseFloat(altura)
28
29          //Atribuindo valores a variável resultado
30          resultado = cacularAreaRetangulo(base, altura)
31
32          //interpolando o resultado na saída
33          document.write('<br>')
34          document.write(`O cálculo da área é ${resultado}m²`)
35
36      </script>
37  </body>
```

→ Escopo Local

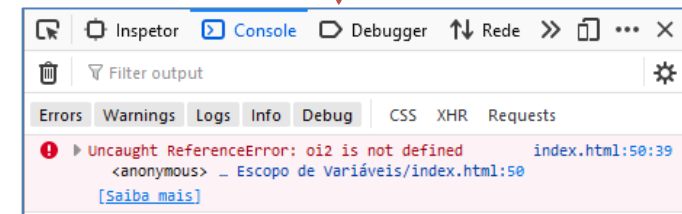
→ Escopo Global

# JavaScript – Escopo de variáveis

```
19 <h1>Escopo de Variáveis</h1>
20 <hr>
21 <h1 class=text-primary>Global</h1>
22
23 <script>
24   document.write('<h3></h3>')
25   //Variável saudacao em escopo Global
26   let oi = 'Oi!'
27   //Função
28   function saudar(){
29     document.write(`Saudação: ${oi}`)
30   }
31   //Chamar a Função
32   saudar()
33   document.write(`<br>Você me disse ${oi}!!!!`)
34 </script>
35 <hr>
36
37
38
39 <h1 class=text-primary>Local</h1>
40 <script>
41   //Função
42   function saudar(){
43     //Declarando uma variável local
44     let oi2 = 'Oi!' //sem a declaração var ou let vira global
45     document.write(`Saudação: ${oi2}`)
46   }
47   //Chamar a Função
48   saudar()
49
50   document.write(`<br>Você me disse ${oi2}!!!!`)
51 </script>
```

→ GLOBAL: EVITAR

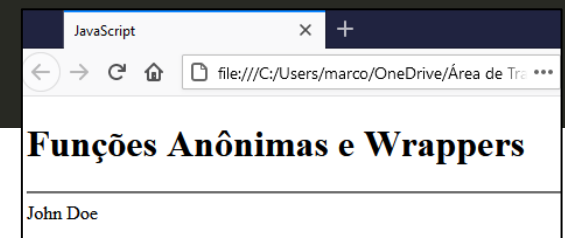
LOCAL: MAIS SEGURO



# JavaScript – Função Anônima e wrappers

Na programação de computadores, uma função anônima é uma definição de função que não está vinculada a um identificador. As funções anônimas geralmente são argumentos passados para funções de ordem superior ou usadas para construir o resultado de uma função de ordem superior que precisa retornar uma função. Como eu vou chamar a função se ela não tem nome definido? Simples... usando *wrappers*, ou seja, podemos associar uma função a uma variável. Esta variável vai guardar uma referência para a função.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>JavaScript</title>
5     <meta charset="utf-8">
6     <link rel="stylesheet" type="text/css" href="estilo.css">
7   </head>
8
9   <body>
10    <h1>Funções Anônimas e Wrappers</h1>
11    <hr>
12    <script>
13
14      //Declarando uma variável com uma função
15      let mostraNome = function(nome) {
16        document.write(nome)
17      }
18
19      //chamndo a função
20      //Note que é uma variável e não uma função
21      mostraNome('John Doe')
22
23    </script>
24  </body>
25 </html>
```



# JavaScript – Função Callback

De forma simples, *callback* é uma função passada como parâmetro para outra função.



```
9      <body>
10      <h1>Funções Callback</h1>
11      <hr>
12
13      <script>
14
15          //Declarando uma função
16          function exibirArtigo(id, callbackSucesso, callbackErro) {
17              //lógica exemplo do tratamento de secesso ou erro
18              if (true) {
19                  callbackSucesso ('Função callback em JS', 'Aprendendo callback!'
20                                )
21              } else { //tratando o erro
22                  callbackErro ('Erro ao recuperar os dados!')
23              }
24          }
25
26          //Criando os wrappers
27          let callbackSucesso = function(titulo, descricao) {
28              document.write('<h2>' + titulo + '</h2>')
29              document.write('<hr>')
30              document.write('<p>' + descricao + '</p>')
31          }
32
33          let callbackErro = function(erro) {
34              document.write('<p><b>Erro:</b> ' + erro + '</p>')
35          }
36
37          //Chamando a função ExibirArtigo
38          //inserindo o ID e os dois wrappers, ou seja, o callback
39          exibirArtigo(1, callbackSucesso, callbackErro)
40      </script>
41  </body>
```



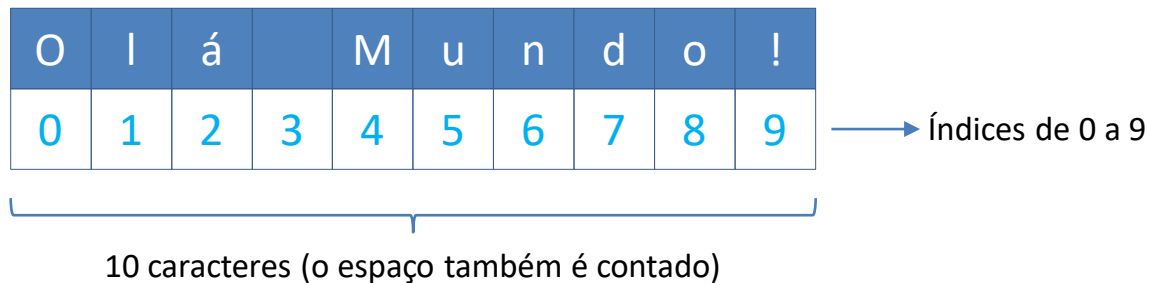
# JavaScript – Funções nativas para trabalhar com Strings

Valores primitivos como strings por si só não possuem propriedades ou métodos.

Mas o JS como sua interpretação pode convertê-los em objetos. Assim podemos acessar propriedades e métodos desses objetos.

Essas funcionalidades se dividem em dois grupos. Ref: ([https://www.w3schools.com/jsref/jsref\\_obj\\_string.asp](https://www.w3schools.com/jsref/jsref_obj_string.asp))

Propriedade `length`: Retorna a quantidade de caracteres de uma `String`



Para encontramos um caractere específico procurando pelo seu índice utilizando o método `charAt()`.

Exemplo:

o caractere 'M' está na posição 4.

o caractere 'd' está na posição 7

o caracteres '!' está na posição 9

**Observação:**

Métodos precisam abrir e fechar parênteses, exemplo: `charAt()`.

# JavaScript – Funções nativas para trabalhar com Strings

Praticando...

```
9      <body>
10      <h1>Propriedade length</h1>
11      <hr>
12
13      <script>
14          //Declarando a variável
15          let texto = 'Olá Mundo!'
16          quant_caractere = texto.length
17
18          //Saída com a quantidade de Caracteres
19          document.write(
20              'A frase <span class="verde">Olá Mundo!</span> possui '
21              + quant_caractere + ' Caracteres')
22
23          //Saltando uma linha
24          document.write('<br>')
25
26          //Encontrando o caractere pelo índice
27          document.write(
28              'Na posição <span class="verde">4</span> temos o caractere '
29              + texto.charAt(4))
```



# JavaScript – Funções nativas para trabalhar com Strings

Para encontramos a primeira ocorrência de um caractere em uma *String* podemos utilizar o método `indexOf()`.

O	I	á		M	u	n	d	o	!
0	1	2	3	4	5	6	7	8	9

→ Índices de 0 a 9

10 caracteres (o espaço também é contado)

Exemplo `indexOf( 'M' )` → Retorno 4

## Observação:

Quando **não houver** uma ocorrência o retorno é -1.  
Há diferença entre maiúsculas e minúsculas.

# JavaScript – Funções nativas para trabalhar com Strings

O método `replace()` pesquisa em uma string por um valor especificado ou uma expressão regular e retorna uma nova string onde os valores especificados são substituídos

O	I	á		M	u	n	d	o	!
0	1	2	3	4	5	6	7	8	9

→ Índices de 0 a 9

10 caracteres (o espaço também é contado)

Exemplo `replace('Olá Mundo!', 'Hello World!!!')`

# JavaScript – Funções nativas para trabalhar com Strings

O método `substr()` extrai os caracteres de uma string, começando em uma posição inicial especificada e até o número especificado de caracteres

O	I	á		M	u	n	d	o	!
0	1	2	3	4	5	6	7	8	9

→ Índices de 0 a 9

10 caracteres (o espaço também é contado)

Exemplo `substr(0, 2)` → Retorno `'Olá'`

## Observação:

Não podemos usar valores negativos.

# JavaScript – Funções nativas para trabalhar com Strings

Podemos mudar as Strings de maiúsculas para minúsculas utilizando os métodos `toUpperCase()` e `toLowerCase()` respectivamente.

O	l	á		M	u	n	d	o	!
0	1	2	3	4	5	6	7	8	9

→ Índices de 0 a 9

10 caracteres (o espaço também é contado)

Exemplo `toUpperCase()` → Retorno `'OLÁ MUNDO!'`

Exemplo `toLowerCase()` → Retorno `'olá mundo!'`

-		J	o	h	n		-		!
0	1	2	3	4	5	6	7	8	9

Para retirarmos os espaços em branco do início e do fim de cadeia de Strings, utilizamos o método `trim()`.

Exemplo `nome.trim()` → Retorno `'- John -!'`

# JavaScript – Funções nativas para trabalhar com Strings

Praticando...

```
30
31 //Saltando uma linha
32 document.write('<br>')
33 //Encontrando 1ª ocorrência de um caractere
34 document.write(
35     'O Caractere <span class="verde">M</span> está posição '
36     + texto.indexOf('M'))
37
38 //Saltando uma linha
39 document.write('<br>')
40 //Substituindo caracteres ou strings em um texto
41 document.write(
42     'A frase <span class="verde">Olá Mundo!</span> em Inglês é '
43     + texto.replace('Olá Mundo!', ' Hello World!!!'))
44
45 //Saltando uma linha
46 document.write('<br>')
47 //Extraíndo partes de uma String
48 document.write(
49     'A frase <span class="verde">Olá Mundo!</span> começa com '
50     + texto.substr(0, 3))
51
52 //Saltando uma linha
53 document.write('<br>')
54 //Frase em MAIÚCULO
55 document.write(
56     'A frase <span class="verde">Olá Mundo!</span> em Maiúsculo '
57     + texto.toUpperCase())
58
59 //Saltando uma linha
60 document.write('<br>')
61 //Frase em minúsculo
62 document.write(
63     'A frase <span class="verde">Olá Mundo!</span> em Minúsculo '
64     + texto.toLowerCase())
```

# JavaScript – Funções nativas para trabalhar com Strings

---

Praticando...

```
65
66      //Saltando uma linha
67      document.write('<br>')
68      //Retirando espaços antes de depois de uma cadeia de strings
69      let texto2 = ' - John - ! '
70      document.write('<br>')
71      document.write('Tamanho da String antes do trim ' + texto2.length)
72      document.write('<br>')
73
74      texto3 = texto2.trim()
75      document.write('A frase sem espaços ' + texto3)
76      document.write('<br>')
77      document.write('Tamanho da String depois do trim ' + texto3.length)
78
79      </script>
80  </body>
81
82  </html>
```



# JavaScript – Funções matemáticas

---

Math é um objeto embutido que tem propriedades e métodos para constantes e funções matemáticas. Não é um objeto de função. Ao contrário de outros objetos globais, Math não é um construtor. Todas as propriedades e métodos de Math são estáticos. Você pode referenciar a constante PI como Math.PI e você pode chamar a função de seno como Math.sin(x), onde x é o argumento do método. Constantes são definidas com a precisão total de números reais em JavaScript.

Mais sobre a Math: [MDN Web Docs](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Math)

Vamos ver alguns exemplos do Math codificados no JS.

1. Math.sqrt – Raiz quadrada
2. Math.pow – Potência
3. Math.trunc – Remove parte fracionária de um número
4. Math.sin – Seno de um ângulo
5. Math.cos – Cosseno de um ângulo
6. Math.tan – Tangente de um ângulo
7. Math.ceil – Arredonda para cima
8. Math.floor – Arredonda para baixo
9. Math.round – Arredondamento com base matemática
10. Math.random – Retorna um número aleatório entre 0 e 1

# JavaScript – Raiz Quadrada

Praticando...

```
9  <body>
10
11  <h1 class="verde">Funções Matemáticas: Raiz Quadrada</h1>
12  <hr>
13
14  <script>
15
16      //Entra com um número via caixa de diálogo
17      let numero = prompt('Entre com um número')
18
19      //Casting
20      numero = parseInt(numero)
21
22      //Atribuindo o valor da raiz para uma variável
23      raizQuadrada = Math.sqrt(numero)
24
25      //Saída com template String
26      document.write(`A raiz quadrada de ${numero} é ${raizQuadrada}`)
27
28  </script>
29
30 </body>
```

# JavaScript – Potência

Praticando...

```
9  <body>
10
11  <h1 class="verde">Funções Matemáticas: Potenciação</h1>
12  <hr>
13
14  <script>
15      //Declarando a variável
16      let base = prompt('Entre com um número da BASE: ')
17      let expoente = prompt('Entre como o EXPOENTE: ')
18
19      //Casting propositalmente para inteiro
20      base = parseInt(base)
21      expoente = parseInt(expoente)
22
23
24      //Calculando a potência
25      potencia = Math.pow(base, expoente)
26
27      //Saída com template String
28      document.write(`0 número ${base} elevado a ${expoente} = ${potencia}`)
29
30  </script>
31
32 </body>
```

# JavaScript – Remover parte fracionária

Praticando...

```
9  <body>
10
11  <h1 class="verde">Funções Matemáticas: Truncando valores</h1>
12  <hr>
13
14  <script>
15      //Declaração e entrada de dados
16      let numeroDecimal = prompt('Entre com um número decimal: ')
17
18      //Casting
19      numeroDecimal = parseFloat(numeroDecimal)
20
21      //retirando a parte fracionária
22      numeroInteiro = Math.trunc(numeroDecimal)
23
24      //Saída com template String
25      document.write(`0 número ${numeroDecimal} truncado é ${numeroInteiro}`)
26
27  </script>
28
29  </body>
```

# JavaScript – Seno, cosseno e tangente

Praticando...

```
9  <body>
10
11  <h1 class="verde">Funções Matemáticas: Seno, Cosseno e Tangente</h1>
12  <hr>
13
14  <script>
15      //Declaração e entrada de dados
16      let angulo = prompt('Entre com o valor do ângulo: ')
17
18      //Casting
19      angulo = parseFloat(angulo)
20
21      //Calculando seno e cosseno
22      seno = Math.sin(angulo * Math.PI / 180)
23      cosseno = Math.cos(angulo * Math.PI / 180)
24      tangente = Math.tan(angulo * Math.PI / 180)
25
26      //Saída com template String
27      document.write(`Ângulo: ${angulo}
28          <br> Seno: ${seno} <br> Cosseno ${cosseno} <br> Tangente: ${tangente}`)
29
30  </script>
31
32  </body>
```

# JavaScript – Arredondar para baixo

Praticando...

```
9  <body>
10
11  <h1 class="verde">Funções Matemáticas: Arredondar para baixo</h1>
12  <hr>
13
14  <script>
15      //Declaração e entrada de dados
16      let numero = prompt('Entre com um valor quebrado: ')
17
18      //Casting
19      numero = parseFloat(numero)
20
21      //Arredondamento
22      arredondamento = Math.floor(numero)
23
24      //Saída com template String
25      document.write(`0 número ${numero} arredondado para baixo é ${arredondamento}`)
26
27  </script>
28
29 </body>
```

# JavaScript – Arredondar para cima

Praticando...

```
9  <body>
10
11  <h1 class="verde">Funções Matemáticas: Arredondar para cima</h1>
12  <hr>
13
14  <script>
15      //Declaração e entrada de dados
16      let numero = prompt('Entre com um valor quebrado: ')
17
18      //Casting
19      numero = parseFloat(numero)
20
21      //Arredondamento
22      arredondamento = Math.ceil(numero)
23
24      //Saída com template String
25      document.write(`0 número ${numero} arredondado para cima é ${arredondamento}`)
26
27  </script>
28
29 </body>
```

# JavaScript – Arredondamento matemático

Praticando...

```
9  <body>
10
11  <h1 class="verde">Funções Matemáticas: Arredondar com base matemática</h1>
12  <hr>
13
14  <script>
15      //Declaração e entrada de dados
16      let numero = prompt('Entre com um número decimal: ')
17
18      //Casting
19      numero = parseFloat(numero)
20
21      //Calculando o arredondamento
22      arredondamento = Math.round(numero)
23
24      //Saída com template String
25      document.write(`O número ${numero} arredondado é ${arredondamento}`)
26
27  </script>
28
29  </body>
```



# JavaScript – Gerar número randômico

Praticando...

```
9  <body>
10
11  <h1 class="verde">Funções Matemáticas: Arredondar com base matemática</h1>
12  <hr>
13
14  <script>
15      //Declarando variável
16      let quant_numeros = 5
17
18      //Atribuindo o valor a uma variável
19      let numero_aleatorio1 = Math.round(Math.random() * quant_numeros)
20      let numero_aleatorio2 = Math.round((Math.random()*quant_numeros)+1)
21
22      //Saída com template String
23      document.write(`0 número sorteado (com o zero) foi: ${numero_aleatorio1}`)
24      document.write('<br>')
25      document.write(`0 número sorteado (sem o zero) foi: ${numero_aleatorio2}`)
26
27  </script>
28
29  </body>
```

# JavaScript – Funções com datas

---

JavaScript não possui dados do tipo data. No entanto, você pode usar o objeto Date e seus métodos para trabalhar com datas e horas nas suas aplicações. O objeto Date tem um grande número de métodos para setar, recuperar e manipular datas. Ele não tem nenhuma propriedade.

JavaScript manipula datas de maneira semelhante ao Java. As duas linguagens tem muitos dos mesmos métodos para lidar com datas e ambas armazenam datas como números em **milissegundos**, desde 1 de janeiro de 1970, às 00:00:00 (January 1, 1970, 00:00:00).

## Data

getDate() – Recupera o dia

getMonth() – Recupera o Mês

getFullYear() – Recupera o ano

## Hora

getHours() – Recupera a hora

getMinutes() – Recupera os minutos

getSeconds() – Recupera os segundos

# JavaScript – Funções com datas: Exemplo

```
9 <body>
10
11 <h1 class="verde">Calcular dias entre datas</h1>
12 <hr>
13
14 <script>
15
16 //Criando o objeto data
17 // 1 de Janeiro
18 let data_1 = new Date(2021, 0, 1)
19 // 5 de Fevereiro
20 let data_2 = new Date(2021, 1, 5)
21
22 //Mostrar as datas de forma textual
23 document.write(`<h1>${data_1.toString()}</h1>`)
24 document.write(`<h1>${data_2.toString()}</h1>`)
25
26 //Transofar essa data em milissegundos
27 document.write(`<h1 class="laranja">Data 1 em milissegundos: ${data_1.getTime()}</h1>`)
28 document.write(`<h1 class="laranja">Data 2 em milissegundos: ${data_2.getTime()}</h1>`)
29
30 //Diferença das datas em milissegundos
31 let tot_milissegundos = Math.abs(data_1.getTime() - data_2.getTime())
32 document.write(`<h1 class="verde">Diferença entre as datas: ${tot_milissegundos}</h1>`)
33
34 //Calculando os milissegundos de 1 dia
35 let milissegundos_dia = 1 * 24 * 60 * 60 * 1000
36
37 //1 dia tem tantos milissegundos
38 document.write(`<h1>1 dia tem ${milissegundos_dia} milissegundos</h1><br>`)
39
40 //Para encontrar a quantidade de dias
41 //Dividimos o total de dias em milissegundos pelos milissegundos de 1 dia
42 dias = tot_milissegundos / milissegundos_dia
43
44 //Saída em template String
45 document.write(`<h1 class="vermelha">De 1/01/2021 a 05/02/2021 tem ${dias} dias!</h1>`)
46
47 </script>
48
49 </body>
```

## Calcular dias entre datas

Fri Jan 01 2021 00:00:00 GMT-0300 (Horário Padrão de Brasília)

Fri Feb 05 2021 00:00:00 GMT-0300 (Horário Padrão de Brasília)

Data 1 em milissegundos: 1609470000000

Data 2 em milissegundos: 1612494000000

Diferença entre as datas: 3024000000

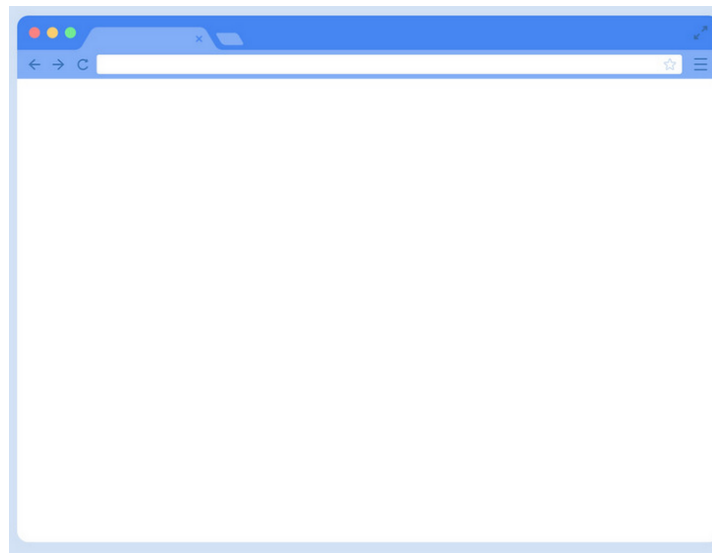
1 dia tem 86400000 milissegundos

De 1/01/2021 a 05/02/2021 tem 35 dias!

# JavaScript: Eventos

---

Faremos inicialmente a apresentação dos fundamentos e da terminologia e, a seguir, examinaremos os manipuladores de eventos e o conceito de propagação de eventos. Serão apresentados o objeto evento e suas respectivas propriedades e seus métodos. Os tipos de eventos serão detalhados e exemplificados, e será apresentado um script crossbrowser para desenvolvimento com uso de eventos.



## **Tipos de eventos:**

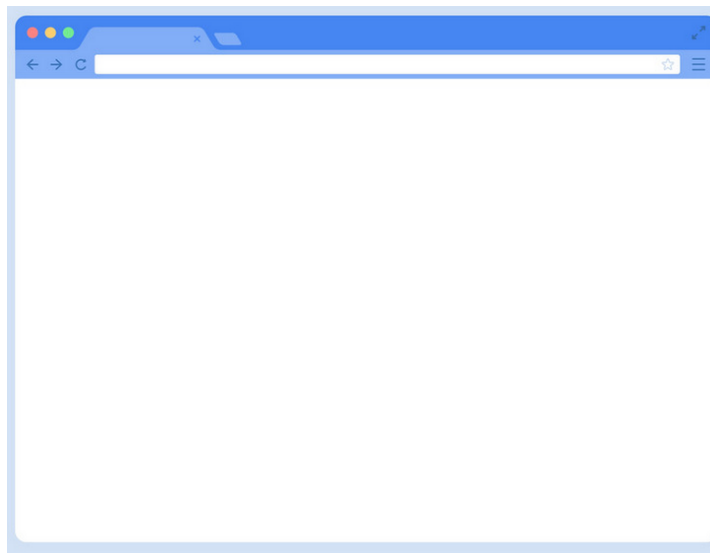
- Mouse
- Janela
- Teclado
- Formulário

# JavaScript: Eventos

---

## Evento?

Em evento ocorre toda vez que há interação com um documento web e assim, um evento é disparado. Carregar ou fechar uma página caracteriza a ocorrência de dois eventos distintos. O clique do usuário em um link e o passar do ponteiro do mouse sobre um menu pop-up caracterizam a ocorrência de eventos. Eventos podem ocorrer por interferência do usuário (clique em um link) ou por interferência do próprio navegador (carregar uma página).



## Tipos de eventos:

- Mouse
- Janela
- Teclado
- Formulário

# JavaScript: Eventos

---

## **Disparador de Evento**

É o elemento HTML ao qual foi atrelado um manipulador de evento.

## **Manipulador de Evento**

Manipulador-padrão de evento O termo default event handler, que em tradução livre significa manipulador-padrão de evento, é uma função-padrão do navegador executada quando ocorrem determinados tipos de eventos.

Por exemplo: clicar em um link dispara uma função-padrão que faz com que o navegador renderize a página definida no link. No entanto, ao movimentar o ponteiro do mouse sobre um parágrafo, nada acontece, visto que não existe uma função-padrão para esse evento.

## **Manipulador de evento no HTML**

Consiste em atrelar a função ao elemento com uso do atributo HTML correspondente ao evento. Por exemplo, para os eventos click e mouseout, os atributos HTML correspondentes são: onclick e onmouseout.

# JavaScript: Eventos do Mouse

---

Eventos acionados com o mouse

- `onclick`: Acionado com Clique
- `ondblclick`: Acionado com duplo clique
- `onmouseup`: Acionado com mouse pressionado
- `onmouseover`: Acionado com o mouse sobre o objeto
- `onmouseout`: Acionado com o mouse saindo do objeto

# JavaScript: Eventos do Mouse: Exemplo

```
1  <!DOCTYPE html>
2  <html lang="pt-br">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <!-- Bootstrap CSS -->
9      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet"
10         integrity="sha384-e0JMYsd53ii+sc0/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6" crossorigin="anonymous">
11
12     <title>JavaScript</title>
13
14     <script>
15         //Eventos do Mouse
16         function eventoClique() {
17             alert('Você clicou uma vez!')
18         }
19         //
20         function eventoDuploClique() {
21             alert('Você clicou 2 vezes!')
22         }
23
24         function eventoSoltarMouse() {
25             alert('Você soltou o mouse!')
26         }
27
28         function eventoSobreMouse() {
29             alert('Você está com o mouse sobre o elemento!')
30         }
31
32         function eventoForaMouse() {
33             alert('Você saiu do elemento!')
34         }
35     </script>
36
37 </head>
38
```



# JavaScript: Eventos do Mouse: Exemplo

```
39 <body>
40   <div class="container">
41     <h1>Eventos do Mouse</h1>
42     <hr>
43     <div class="row">
44       <div class="col-2">
45         <div class="bg-primary p-4" onclick="eventoClique()">
46           <p class="text-center">Ação com 1 clique do mouse</p>
47         </div>
48       </div>
49
50       <div class="col-2">
51         <div class="bg-info p-4" ondblclick="eventoDuploClique()">
52           <p class="text-center">Ação com 2 cliques do mouse</p>
53         </div>
54       </div>
55
56       <div class="col-2">
57         <div class="bg-danger p-4" onmouseup="eventoSoltarMouse()">
58           <p class="text-center">Ação soltar o botão do mouse </p>
59         </div>
60       </div>
61
62       <div class="col-2">
63         <div class="bg-success p-4" onmouseover="eventoSobreMouse()">
64           <p class="text-center">Ação mouse sobre o elemento</p>
65         </div>
66       </div>
67
68       <div class="col-2">
69         <div class="bg-warning p-4" onmouseout="eventoForaMouse()">
70           <p class="text-center">Ação mouse sai do elemento</p>
71         </div>
72       </div>
73
74       <div class="col-2">
75         <div class="bg-light p-4" onmouseover="eventoSobreMouse()" onmouseout="eventoForaMouse()">
76           <p class="text-center">Ação sobre e fora do elemento</p>
77         </div>
78       </div>
79     </div>
80   </div>
81
```

# JavaScript: Eventos do Mouse: Exemplo

```
82     <!-- Option 1: Bootstrap Bundle with Popper -->
83     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js"
84         integrity="sha384-JEW9xMcG8R+pH31jmWH6WP0WintQrMb4s7Z0dauHnUtxwog2vI5DkLts3qm9Ekf" crossorigin="anonymous">
85     </script>
86     <!--
87     <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.1/dist/umd/popper.min.js" integrity="sha384-SR1sx49pcuLnqZUnnPwa
88     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.min.js" integrity="sha384-j0CNLUeiqtyaRmlzU
89     -->
90
91 </body>
92
93 </html>
```

# JavaScript: Eventos de Janela

---

Esses eventos ocorrem quando fazemos interações com o a janela do browser, ora redimensionando a janela, ora utilizando o botão de rolagem (o botão do meio do mouse). E para essas ações são utilizadas as seguintes propriedades:

- `onscroll`: Esse evento é acionado quando utilizando o botão de rolagem do mouse no browser.
- `onresize`: Esse evento é acionado quando a página é redimensionada

# JavaScript: Eventos de Janela – Exemplo onresize

```
16  <!--Esse evento é específico do Body-->
17
18  <body onresize="console.log('Evento Funcionou')">
19      <div class="container">
20          <h1 class="display-6">Eventos de Janela ONRESIZE</h1>
21          <hr>
22          <div style="height: 500px; width:200px; background: #999;"></div>
23
24          <!--
25              CONSULTAR A DOCUMENTO W3C PARA SE INFORMAR DE TODOS
26              OS EVENTOS E SEUS RESPECTIVOS ELEMENTOS, POIS NEM TODO
27              ELEMENTO TEM SUPORTE A UM EVENTO ESPECÍFICO.-->
28
29      </div>
30
31      <!-- Option 1: Bootstrap Bundle with Popper -->
32      <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js"
33          integrity="sha384-JEW9xMcG8R+pH31jmWH6WP0WintQrMb4s7ZOdauHnUtxwoG2vI5DkLtS3qm9Ekf" crossorigin="
34      </script>
35      <!--
36      <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.1/dist/umd/popper.min.js" integrity="sha384-
37      <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.min.js" integrity="sha3
38      -->
39  </body>
```

# JavaScript: Eventos de Janela – Exemplo onscroll

```
16 <!--Esse evento é específico do Body-->
17
18 <body onscroll="console.log('Evento Funcionou')">
19   <div class="container">
20     <h1 class="display-6">Eventos de Janela ONSCROLL</h1>
21     <hr>
22     <div style="height: 5000px; width:200px; background: #999;"></div>
23
24     <!--
25     CONSULTAR A DOCUMENTO W3C PARA SE INFORMAR DE TODOS
26     OS EVENTOS E SEUS RESPECTIVOS ELEMENTOS, POIS NEM TODO
27     ELEMENTO TEM SUPORTE A UM EVENTO ESPECÍFICO.-->
28
29   </div>
30
31   <!-- Option 1: Bootstrap Bundle with Popper -->
32   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js"
33     integrity="sha384-JEW9xMcG8R+pH31jmWH6WP0WintQrMb4s7Z0dauHnUtxwoG2vI5DkLtS3qm9Ekf" crossorigin="anonymous">
34   </script>
35   <!--
36   <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.1/dist/umd/popper.min.js" integrity="sha384-SR1sx49pcuLn
37   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.min.js" integrity="sha384-j0CNLUeig
38   -->
39 </body>
```

# JavaScript: Eventos do Teclado

---

Os eventos de teclado descrevem uma interação do usuário com o teclado; cada evento descreve uma única interação entre o usuário e uma tecla (ou combinação de uma tecla com teclas modificadoras) no teclado.

**Observação:** os eventos de teclado apenas indicam qual interação o usuário teve com uma tecla em um nível baixo, não fornecendo nenhum significado contextual para essa interação. Quando você precisar lidar com a entrada de texto, use o input. **Os eventos de teclado não podem ser disparados se o usuário estiver usando um meio alternativo de inserir texto**, como um sistema de escrita à mão em um tablet ou tablet gráfico.

Eventos:

- `onkeydown`: Esse evento é disparado quando se usa uma tecla.
- `onkeypress`: Esse evento é disparado quando se pressiona uma tecla (somente caracteres).
- `onkeyup`: Esse evento é disparado quando soltamos a tecla.

# JavaScript: Eventos do Teclado - Exemplo

```
4 <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <!-- Bootstrap CSS -->
9   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet"
10   |   integrity="sha384-e0JMYsd53ii+sc0/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rpP48ckxlpbzkGwra6" crossorigin="anonymous">
11   <title>JavaScript</title>
12
13   <script>
14     //Eventos do teclado
15     function evento1(a, b) {
16       alert(`A soma é ${a + b}! Evento onkeydown disparado`)
17     }
18     function evento2() {
19       alert('Evento onkeyup disparado!')
20     }
21     function evento3() {
22       alert('Evento onkeypress disparado!')
23     }
24   </script>
25
26 </head>
27
28 <body>
29   <div class="container">
30     <h1 class="display-5">Eventos do Teclado</h1>
31     <hr>
32     <!--ONKEYDOWN: É acionado quando pressiono qualquer tecla-->
33     <!--ONKEYUP: É acionado quando uma tecla é liberada-->
34     <!--ONKEYPRESS: É acionado quando matenho a tecla pressionada (a tecla tem que ser um caractere)-->
35     <input onkeydown="evento1(5,6)" type="text" class="form-control mb-2 bg-primary" />
36     <input onkeyup="evento2()" type="text" class="form-control mb-2 bg-success" />
37     <input onkeypress="evento3()" type="text" class="form-control bg-warning" />
38   </div>
39
```

# JavaScript: Eventos de Formulário

---

Os eventos de formulários são um conjunto de scripts carregados pela API de Formulários, os quais são desenvolvidos utilizando Javascript e são chamados durante a execução de ação em formulários ou em momentos específicos de interação em formulários.

- `onfocus`: O elemento vai receber o foco do cursor
- `onblur`: Evento acontece quando o elemento perde o foco
- `onchange`: Evento acontece quando o estado é modificado, ex.: `select`



# JavaScript: Eventos de Formulários - Exemplo

```
16 <div class="container">
17
18   <h1 class="display-6">Eventos de Formulários</h1>
19   <hr>
20
21   <div class="form-floating mb-3">
22     <input type="foco" class="form-control" placeholder=""
23     onfocus="console.log('Acionou o Foco!')"
24     onblur="console.log('Perdeu o Foco')">
25     <label>Clique aqui!</label>
26   </div>
27
28   <div class="form-floating">
29     <select class="form-select" onchange="alert('Você escolheu alguma coisa!')">
30       <option selected>Esolha um Estado</option>
31       <option value='1'>Minas Gerais</option>
32       <option value="2">Rio de Janeiro</option>
33       <option value="3">São Paulo</option>
34     </select>
35     <label for="floatingSelect">Abra a caixa de seleção</label>
36   </div>
37
38 </div>
```

# JavaScript: Introdução ao DOM

---

**DOM** é uma API criada com a finalidade de auxiliar o desenvolvimento de aplicações de natureza geral. Uma API é um conjunto padronizado de funções, rotinas, métodos, classes, protocolos e procedimentos gerais destinados a fornecer funcionalidades a serem usadas por softwares ou programas com a finalidade de simplificar a tarefa de desenvolvimento.

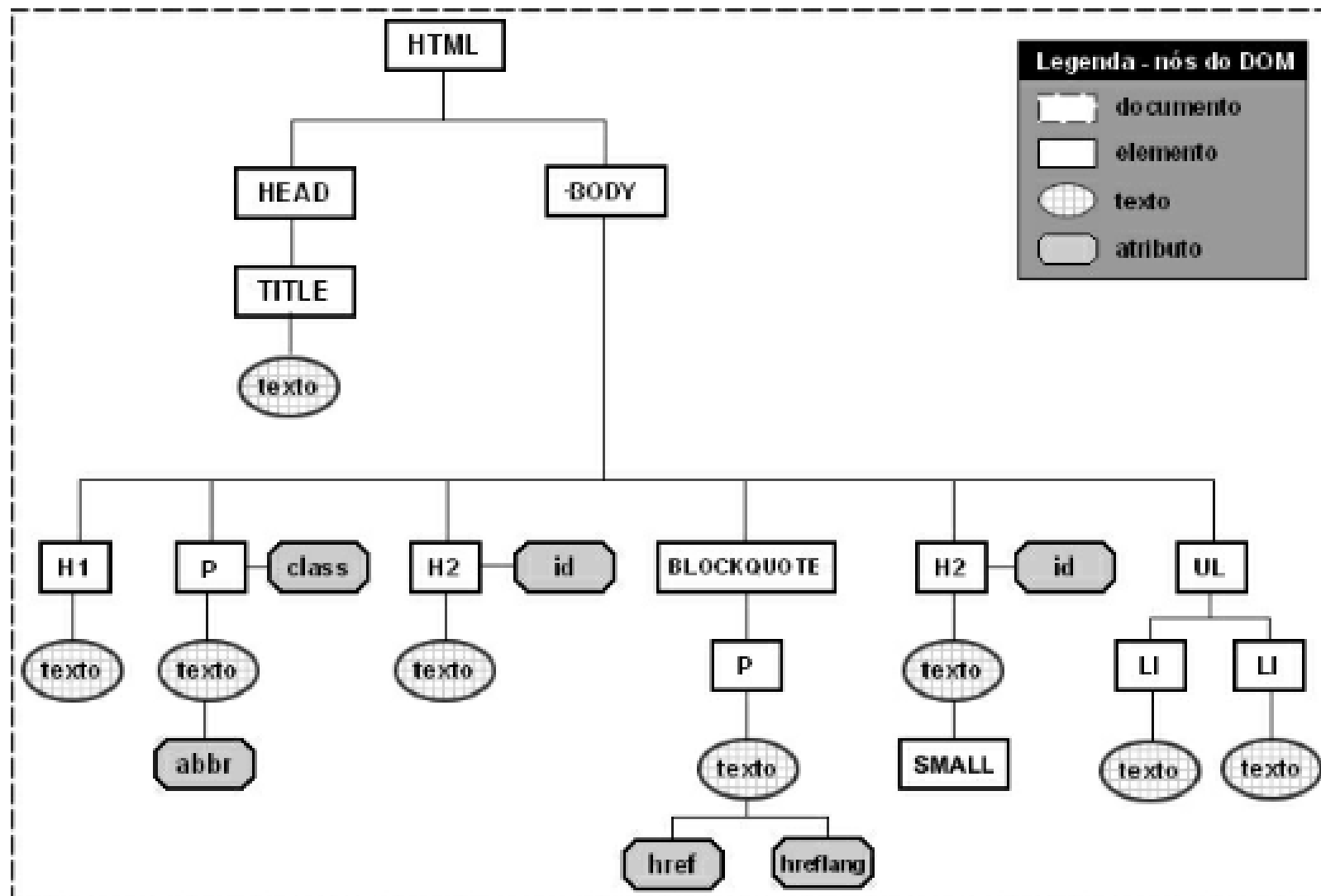
De acordo com a definição geral e para os propósitos específicos da programação com a linguagem JavaScript, podemos dizer que **o DOM é uma API que padroniza a estrutura de documentos HTML e XML, simplificando a tarefa de se acessar e manipular tais documentos. O DOM fornece aos programadores maneiras simples de acessar a estrutura, criar, modificar, adicionar, retirar e manipular elementos e conteúdos de documentos HTML e XML.**

Exemplo à seguir:

# JavaScript: Exemplo

```
1  <!DOCTYPE html>
2  <html lang="pt-br">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <title>DOM</title>
9  </head>
10
11 <body>
12     <h1>Introdução</h1>
13     <p class="um">O DOM é uma API (Application Programming Interface),
14         Interface de Programação para Aplicativos, destinada a documentos escritos com marcação
15         <abbr title="HyperText Markup language">HTML</abbr> ou XML. </p>
16     <h2 id="def">Definição</h2>
17
18     <blockquote>
19         <p> O DOM - Document Object Model do <a href="http://www.w3.org/" hreflang="en">W3C</a> é uma interface
20             independente de plataforma e linguagem que permite os programas e scripts acessar e atualizar dinamicamente
21             a estrutura, o conteúdo e a estilização de documentos. </p>
22     </blockquote>
23
24     <h2 id="tipo">Tipos de <small>DOM</small></h2>
25     <ul>
26         <li>DOM para HTML</li>
27         <li>DOM para XML</li>
28     </ul>
29
30 </body>
31
32 </html>
```

# JavaScript: árvore DOM



# JavaScript: Seleção de Elementos

---

Como todos os nossos elementos estão contidos na árvore do DOM precisamos acessá-los de alguma forma. Assim utilizaremos alguns comandos para fazer esses acessos:

## `getElementById()`

O método acessa o elemento do DOM cujo atributo ID foi definido no parâmetro `id` e retorna uma referência ao elemento com esse ID. Lembre-se de que um determinado valor de ID deve ser único no documento, portanto o retorno desse método é um só elemento do DOM.

## `getElementsByTagName()`

O método acessa todos os elementos do DOM do tipo definido no parâmetro `tag` e retorna um array cujos itens fazem referência aos elementos desse tipo, na ordem em que aparecem na marcação.

## `getElementsByName()`

O método acessa todos os elementos do DOM cujo atributo `name` tenha sido definido no parâmetro `nome` e retorna um array cujos itens fazem referência aos elementos com o atributo `name` igual a `nome`, na ordem em que aparecem na marcação.

## `getElementsByClassName()`

Seleciona os objetos html a partir de um atributo *name*.

## `innerHTML`

A propriedade `innerHTML` permite que se inspecione ou se defina o conteúdo HTML de um elemento do DOM. Essa propriedade não faz parte das especificações do W3C, mas por ser bem suportada pelos navegadores, pode ser usada. Note que se inspeciona ou insere em um elemento não somente texto, mas também marcação HTML.

# JavaScript: Seleção de Elementos - Exemplo

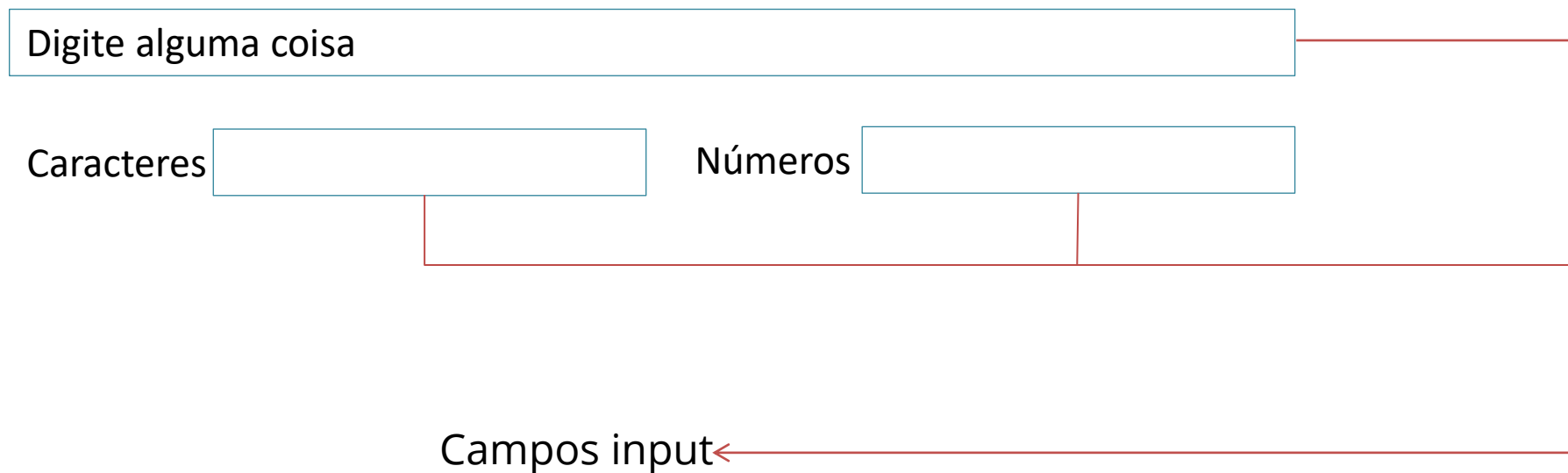
```
17 <body>
18   <div class="container">
19     <h1 class="text-primary">Selecionando Elementos do DOM</h1>
20     <hr>
21     <label class="form-label">Digite seu nome completo</label>
22     <input type="text" id="nome_unico" class="texto_azul form-control" name="nome_cliente" />
23     <label class="form-label">Digite o endereço</label>
24     <input type="text" id="endereco" class="endereco_azul form-control" name="endereco_cliente" />
25     <label class="form-label">Digite o seu cpf</label>
26     <input type="text" id="cpf" class="cpf_azul form-control" name="endereco_cliente" />
27     <label class="form-label">Digite o seu CEP</label>
28     <input type="text" id="cep" class="cpf_azul form-control" name="cep" />
29
30     <script>
31       //Seleção pelo ID
32       console.log(document.getElementById('nome_unico'))
33
34       //Seleção pelo TAGName
35       console.log(document.getElementsByTagName('input'))
36
37       //Seleção pelo ClassName
38       console.log(document.getElementsByClassName('cpf_azul'))
39
40       //Seleção pelo Name
41       console.log(document.getElementsByName('cep'))
42     </script>
43
44     <!-- Option 1: Bootstrap Bundle with Popper -->
45     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/js/bootstrap.bundle.min.js"
46       integrity="sha384-b5kHyXgcpbZJO/tY9U17kGkF150CWuKcCD38l8YkeH8z8QjE0GmW1gYU5S9F0nJ0" crossorigin="anonymous">
47     </script>
48   </div>
49
50 </body>
```

# JavaScript: Alteração de Elementos

---

Na aula anterior vimos como selecionar os elementos do DOM. Nessa aula iremos manipular esses elementos para produzirmos saídas dinâmicas no browser. Começaremos com a alteração de texto dos campos *input*.

Entenda lógica a seguir:



# JavaScript: Alteração de elementos - Exemplo

```
15 <script>
16   //Criando a função separarCaratere()
17   function separarCaracter() {
18     //Colocar esse caractere dentro de uma variável
19     let digito = document.getElementById('character').value
20
21     //Limpar o campo de digitação
22     document.getElementById('character').value = ''
23
24     //Limpar os espaços em branco antes e depois do caractere
25     digito.trim()
26
27     //Condicional
28     switch (digito) {
29       //Como a entrada é uma string precisamos tratá-la como tal
30       case '0':
31       case '1':
32       case '2':
33       case '3':
34       case '4':
35       case '5':
36       case '6':
37       case '7':
38       case '8':
39       case '9':
40         //Colocar o número no local indicado
41         document.getElementById('numero').value += digito
42         break
43       default:
44         //Colocar a letra no local indicado
45         document.getElementById('letra').value += digito
46     }
47   }
48
49   function limpar() {
50     //Limpar o campo de digitação
51     document.getElementById('letra').value = ''
52     document.getElementById('numero').value = ''
53   }
54 </script>
```



# JavaScript: Alteração de elementos - Exemplo

```
65 <body>
66   <div class="container">
67     <h1 class="text-primary">Alterando os valores do elementos do DOM</h1>
68     <hr>
69     <!--
70     |   Vamos capturar uma tecla com evento ONKEYUP, estudado anteriormente
71     |   dentro do input, pois ele pega todos os caracteres.
72     |   E junto com ONKEYUP, chamar uma função para realiar esse evento
73     |-->
74
75     <div class="input-group mb-3">
76       <div class="input-group-prepend">
77         <span class="input-group-text" id="basic-addon1">Insira qualquer texto</span>
78       </div>
79       <input type="text" class="form-control"
80         placeholder="Digite alguma coisa"
81         onkeyup="separarCaracter()" id="caracter">
82     </div>
83
84     <div class="input-group mb-3">
85       <div class="input-group-prepend">
86         <span class="input-group-text">Letras</span>
87       </div>
88       <input type="text" class="form-control" placeholder="Digite alguma coisa" disabled="disabled" id="letra">
89     </div>
90
91     <div class="input-group mb-3">
92       <div class="input-group-prepend"> <span class="input-group-text">Números</span>
93       </div>
94       <input type="text" class="form-control" placeholder="Digite alguma coisa" disabled="disabled" id="numero">
95     </div>
96
97     <label class="btn btn-dark" onclick="limpar()">Limpar</label>
98
```

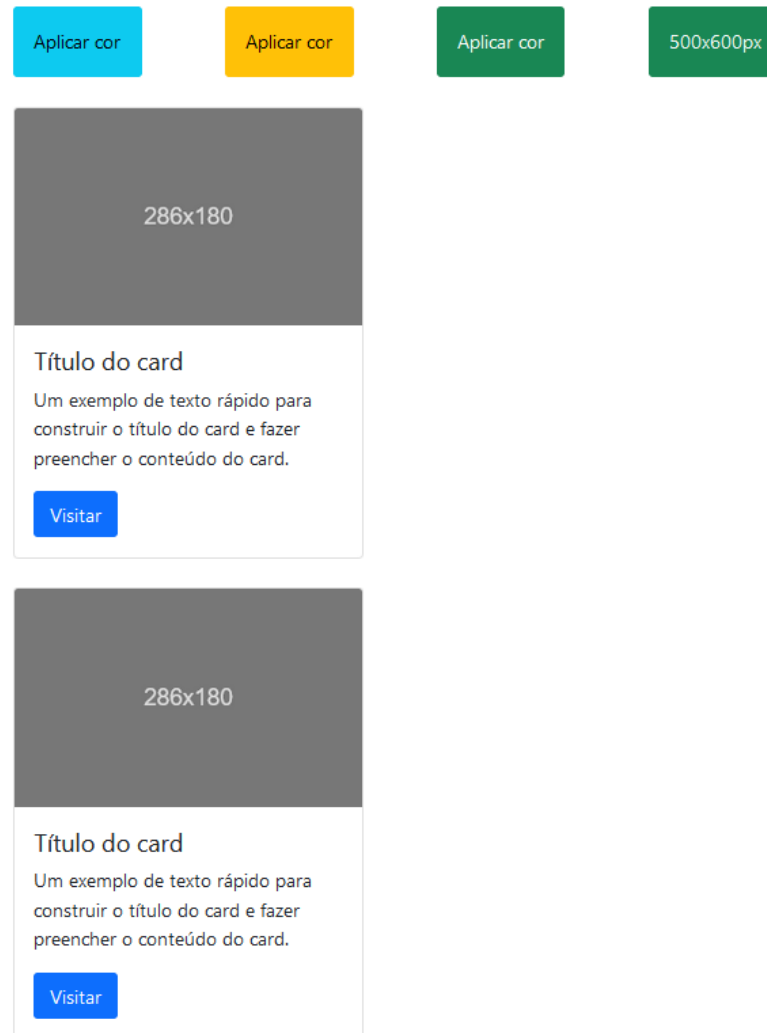
# JavaScript: Manipulando estilos

Vamos utilizar mais um exemplo prático para exercitar a compreensão de seleção e manipulação de elementos utilizando eventos e funções em JavaScript.

O exemplo ao lado mostra como podemos alterar a cor e o tamanho dos elementos do Card Bootstrap

Veja o código a seguir

## Tratando estilos dos elementos



# JavaScript: Manipulando estilos

```
1 <!doctype html>
2 <html lang="pt-br">
3
4 <head>
5   <!-- Required meta tags -->
6   <meta charset="utf-8">
7   <meta name="viewport" content="width=device-width, initial-scale=1">
8
9   <!-- Bootstrap CSS -->
10  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/css/bootstrap.min.css" rel="stylesheet"
11    integrity="sha384-BmbxuPwQa2lc/FVzBcNJ7UAyJxM6wuuqIj61tLrc4wSX0szH/Ev+nYRRuWlolflfl" crossorigin="anonymous">
12
13  <title>JavaScript</title>
14
15  <script>
16    function mudarCor(bg_retangulo) {
17      //Selecionar o elemento: getElementById
18      document.getElementById('areaTexto').style.backgroundColor = bg_retangulo
19
20    }
21
22    function mudarFormato(largura, altura) {
23      document.getElementById('largura').style.width = largura
24      document.getElementById('altura').style.width = largura
25    }
26  </script>
27
28 </head>
29
```

# JavaScript: Manipulando estilos

```
30 <body>
31
32 <div class="container">
33
34 <h1 class="text-success">Tratando estilos dos elementos</h1>
35 <hr>
36
37 <div class="container">
38
39 <div class="row">
40 <div class="col-sm">
41 <button onclick="mudarCor('#31D2F2')" type="button" class="btn btn-info p-3">Aplicar cor</button>
42 </div>
43 <div class="col-sm">
44 <button onclick="mudarCor('#FFCA2C')" type="button" class="btn btn-warning p-3">Aplicar cor</button>
45 </div>
46 <div class="col-sm">
47 <button onclick="mudarCor('#157347')" type="button" class="btn btn-success p-3">Aplicar cor</button>
48 </div>
49 <div class="col-sm">
50 <button onclick="mudarFormato('500px', '600x')" type="button"
51 class="btn btn-success p-3">500x600px</button>
52 </div>
53 </div>
54
55
56 <div class="card mt-4" style="width: 18rem;" id="largura">
57 
58 <div class="card-body" id="areaTexto">
59 <h5 class="card-title">Título do card</h5>
60 <p class="card-text">Um exemplo de texto rápido para construir o título do card e fazer preencher o
61 conteúdo do card.</p>
62 <a href="#" class="btn btn-primary">Visitar</a>
63 </div>
64 </div>
65
```

# JavaScript: Manipulando estilos

```
66     <div class="card mt-4" style="width: 18rem;" id="largura">
67         
68         <div class="card-body" id="areaTexto">
69             <h5 class="card-title">Título do card</h5>
70             <p class="card-text">Um exemplo de texto rápido para construir o título do card e fazer preencher o
71                 conteúdo do card.</p>
72             <a href="#" class="btn btn-primary">Visitar</a>
73         </div>
74     </div>
75
76     <script>
77
78     </script>
79
80     <!-- Option 1: Bootstrap Bundle with Popper -->
81     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/js/bootstrap.bundle.min.js"
82         integrity="sha384-b5kHyXgcpbZJO/tY9U17kGkF150CWuKcCD3818YkeH8z8QjE0GmW1gYU5S9FOnJ0"
83         crossorigin="anonymous">
84     </script>
85
86 </div>
87
88 </body>
89
90 </html>
```

# JavaScript: Manipulando estilos de classes

Vamos utilizar mais um exemplo prático para exercitar a compreensão de seleção e manipulação de elementos utilizando eventos e funções em JavaScript.

Nosso exemplo mais uma vez mostra como fazer modificações no estilo visual dos elementos, agora manipularemos os atributos de classes para alterar a formatação do título e do texto.

Ao lado temos o antes e depois da alteração do estilo via programação em JavaScript

Foram utilizados os eventos `onClick()`, seleção `getElementById` em conjunto com `className`.

Antes

## Tratando classes

### Impsum Lorem

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s.

Alterar a Classe do texto acima

Alterar a Classe do Título

Depois

## Tratando classes

### Impsum Lorem

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s.

Alterar a Classe do texto acima

Alterar a Classe do Título

# JavaScript: Manipulando de estilos - Exemplo

```
15 <style>
16     .estiloTexto {
17         padding: 10px;
18         background: #F7F4F4;
19         font-family: 'Courier New', Courier, monospace;
20         font-weight: bold;
21         font-size: large;
22         color: blue;
23     }
24     .novoEstiloTexto {
25         padding: 20px;
26         background: #9cc0c5;
27         font-family: Georgia, 'Times New Roman', Times, serif;
28         font-weight: normal;
29         font-size: medium;
30         color: white;
31     }
32     .estiloTitulo {
33         font-family: Georgia, 'Times New Roman', Times, serif;
34         font-weight: normal;
35         font-size: medium;
36         color: red;
37     }
38     .novoEstiloTitulo {
39         font-family: Georgia, 'Times New Roman', Times, serif;
40         font-size: x-large;
41         color: black;
42     }
43 </style>
44
45 <script>
46     //Criando a função do evento onclick()
47     function alterarTexto() {
48         document.getElementById('texto').className = 'novoEstiloTexto'
49     }
50
51     function alterarTitulo() {
52         document.getElementById('titulo').className = 'novoEstiloTitulo'
53     }
54 </script>
```

# JavaScript: Manipulando de estilos - Exemplo

```
58 <body>
59
60   <div class="container">
61
62     <h1 class="text-success">Tratando classes</h1>
63     <hr>
64     <div id="texto" class="estiloTexto">
65       <p id="titulo" class="estiloTitulo"><strong>Impsum Lorem</strong></p>
66       <p>
67         Lorem Ipsum is simply dummy text of the
68         printing and typesetting industry. Lorem Ipsum has been the industry's
69         standard dummy text ever since the 1500s.</p>
70     </div>
71
72     <button class="btn btn-success mt-2"
73       type="button"
74       onclick="alterarTexto()">
75       Alterar a Classe do texto acima
76     </button>
77
78     <button class="btn btn-info mt-2"
79       type="button"
80       onclick="alterarTitulo()">
81       Alterar a Classe do Título
82     </button>
83
84     <!-- Option 1: Bootstrap Bundle with Popper -->
85     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/js/bootstrap.bundle.min.js"
86       integrity="sha384-b5kHyXgcpbZJ0/tY9U17kGkF1S0CWuKcCD38l8YkeH8z8QjE0GmW1gYU5S9F0nJ0" crossorigin="anonymous">
87     </script>
88
89   </div>
90
91 </body>
```



# JavaScript: Atividades

---

9. Crie uma página em html (formatação livre) com um campo `<input type="text">` e trabalhe os eventos de acordo com as especificações:

- No evento de **foco** modifique o background do input para Verde.
- Quando o campo **perder o foco**, recupere o seu respectivo valor e faça:
  - Se o conteúdo contido no campo tiver **menos de 3 caracteres** o mesmo deve ter seu background alterado para Amarelo.
  - Se o conteúdo contido no campo tenha **3 caracteres ou mais** o background deve ser alterado para Azul.

Os arquivos deverão ser enviados para:

[sebastiao.silva@mg.senac.br](mailto:sebastiao.silva@mg.senac.br)

Assunto: <seu nome \_ sobrenome:> Atividade DOM

Exemplo: **Sebastião Marcos: Atividade 8 Eventos**

# JavaScript: Atividades

---

10. Utilizando todos os recursos de eventos e seletores, crie um programa javascript para encontrar o resultado de uma equação do segundo grau, dada por  $ax^2 + bx + c = 0$ . Deve-se usar para entradas e saída elementos html. Capriche nos recursos visuais.

Os arquivos deverão ser enviados para:

[sebastiao.silva@mg.senac.br](mailto:sebastiao.silva@mg.senac.br)

Assunto: <seu nome \_ sobrenome:> Atividade DOM

Exemplo: **Sebastião Marcos: Atividade 9 Equação 2º grau**

# JavaScript: innerHTML

```
14 <body>
15   <div class="container">
16
17     <h1 class="display-6">Exemplo innerHTML</h1>
18     <hr>
19
20     <div class="card" style="width: 18rem;">
21       
22       <div class="card-body">
23         <h5 class="card-title">Card title</h5>
24         <p id="texto-card" class="card-text lead">Some quick example text to build on the c
25         bulk of the card's content.</p>
26         <a onclick="alterarTextoP()" href="#" class="btn btn-primary">Português</a>
27         <a onclick="alterarTextoI()" href="#" class="btn btn-primary">Inglês</a>
28       </div>
29     </div>
30
31     <script>
32       function alterarTextoP() {
33         let traducao =
34           `<p class="lead">Algum texto de exemplo rápido para desenvolver o título do
35           <strong>cartão</strong> e compor a maior parte do conteúdo do cartão.</p>`
36         let texto = document.getElementById('texto-card').innerHTML = traducao
37       }
38
39       function alterarTextoI() {
40         let traducao =
41           `Some quick example text to build on the
42           card title and make up the bulk of the cards content.`
43         let texto = document.getElementById('texto-card').textContent = traducao
44       }
45     </script>
46
47 </body>
```

## Exemplo innerHTML



### Card title

Some quick example text to build on the card title and make up the bulk of the card's content.

[Português](#)[Inglês](#)

# Função Eval()

---

Função **eval()**: é uma função de propriedade do objeto global (window).

O argumento da função **eval()** é uma string. Se a string representa uma expressão, **eval()** avalia a expressão. Se o argumento representa uma ou mais declarações em JavaScript, **eval()** avalia as declarações. **Não chame o eval() para avaliar uma expressão aritmética**; JavaScript avalia expressões aritméticas automaticamente.

Se você construir uma expressão aritmética como uma string, você pode usar eval() para calcular o resultado depois. Por exemplo, suponha que você tenha uma variável x. Você pode adiar a avaliação de uma expressão envolvendo x atribuindo o valor de string da expressão, dizer "3 \* x + 2", a uma variável, e, em seguida, chamando **eval()** em um ponto posterior no seu script.

## Eval

---

Entre com uma expressão matemática

1 + 1

Saída

2

Calcular

# Atividade 11: Calculadora

```
14 <body>
15   <div class="container">
16     <h1 class="display-6">Eval</h1>
17     <hr>
18     <label class="form-label">Entre com uma expressão matemática</label>
19     <input id="entrada" class="form-control" type="text">
20     <label class="form-label">Saída</label>
21     <input id="saida" class="form-control" type="text" placeholder="Resultado">
22     <button onclick="calcular()" class="btn btn-primary mt-2">Calcular</button>
23     <script>
24       function calcular() {
25         entrada = document.getElementById('entrada').value
26         saida = eval(entrada)
27         document.getElementById('saida').value = saida
28       }
29     </script>
30     <!-- Option 1: Bootstrap Bundle with Popper -->
31     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/js/bootstrap.bundle.min.js"
32       integrity="sha384-b5kHyXgcpbZJO/tY9U17kGkf1S0CWuKcCD38l8YkeH8z8QjE0GmW1gYU5S9FOnJ0" crossorigin="anonymous">
33     </script>
34   </div>
35 </body>
```

# Atividade 11: Calculadora

---

Utilizando os recursos aprendidos até o momento, **se possível**, construa a calculadora ao lado com suas respectivas funcionalidades. Não precisa ser igual ao exemplo ao lado pode ter formato e cores diferentes, **mas somente com essas funcionalidades matemáticas visualizadas no exemplo, ou seja, soma, produto, subtração e divisão.**

Calculadora Js, CSS e Bootstrap

---

