

# 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/>

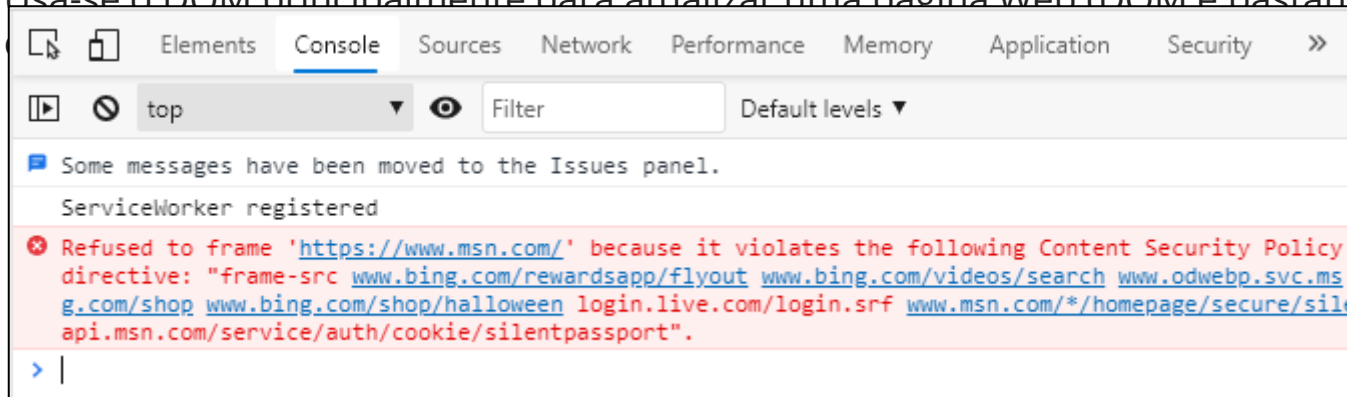
# JavaScript

## 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.

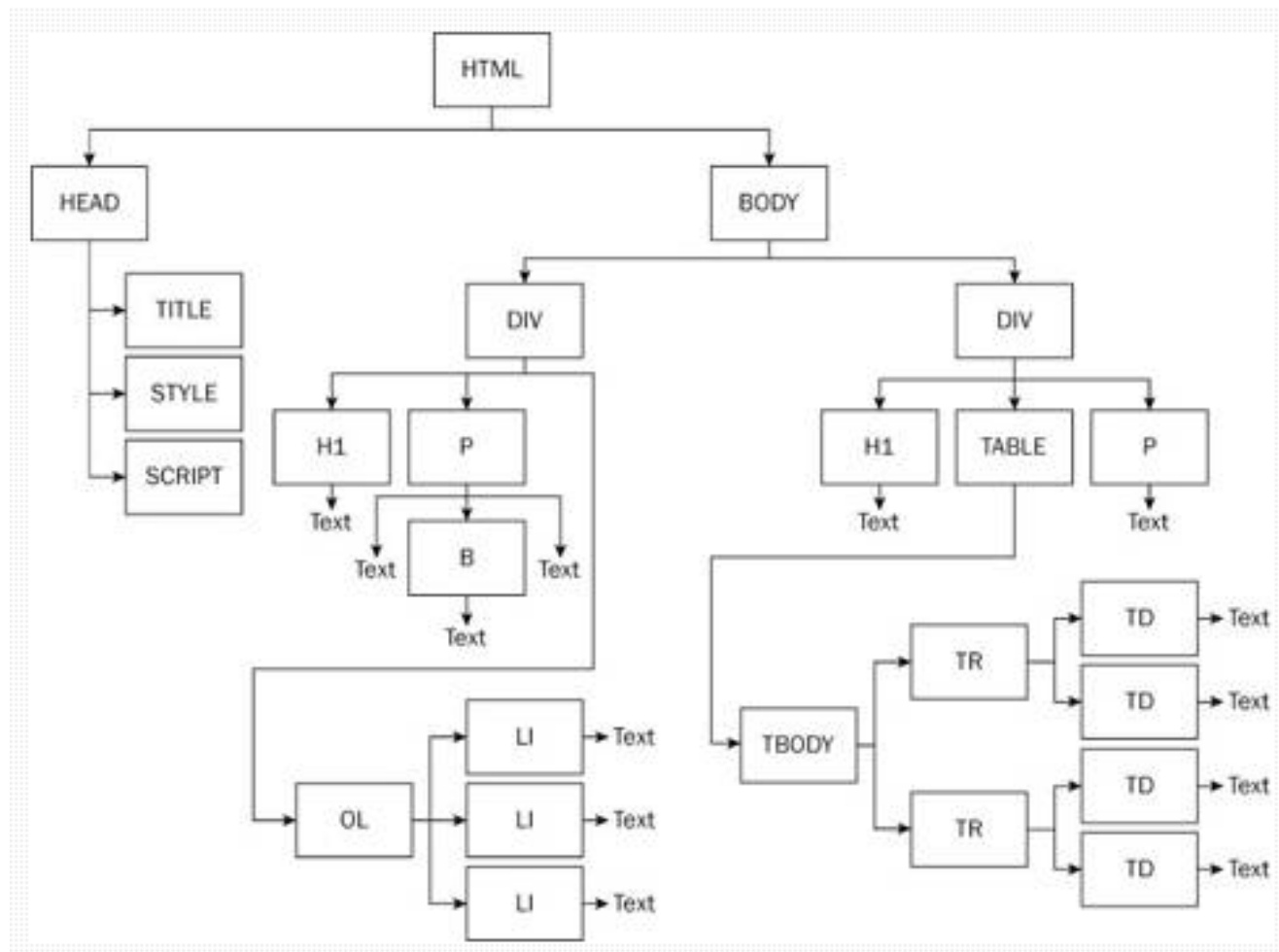
Usa-se o DOM principalmente para atualizar uma página Web (DOM é bastante utilizado com Ajax) **Para o DOM aperte F12.**





# 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>`

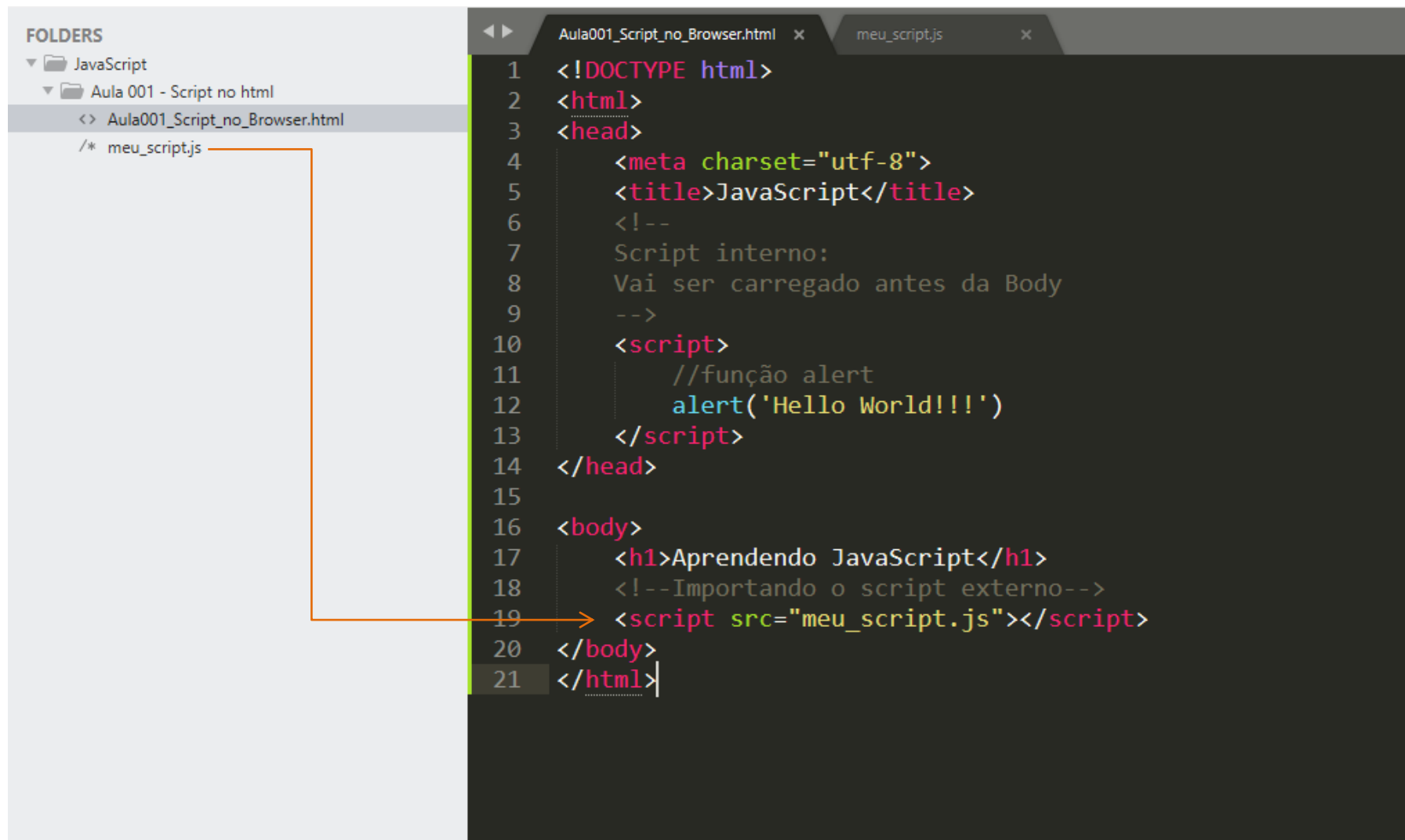
Para essa segunda opção devemos seguir algumas regras:

- 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



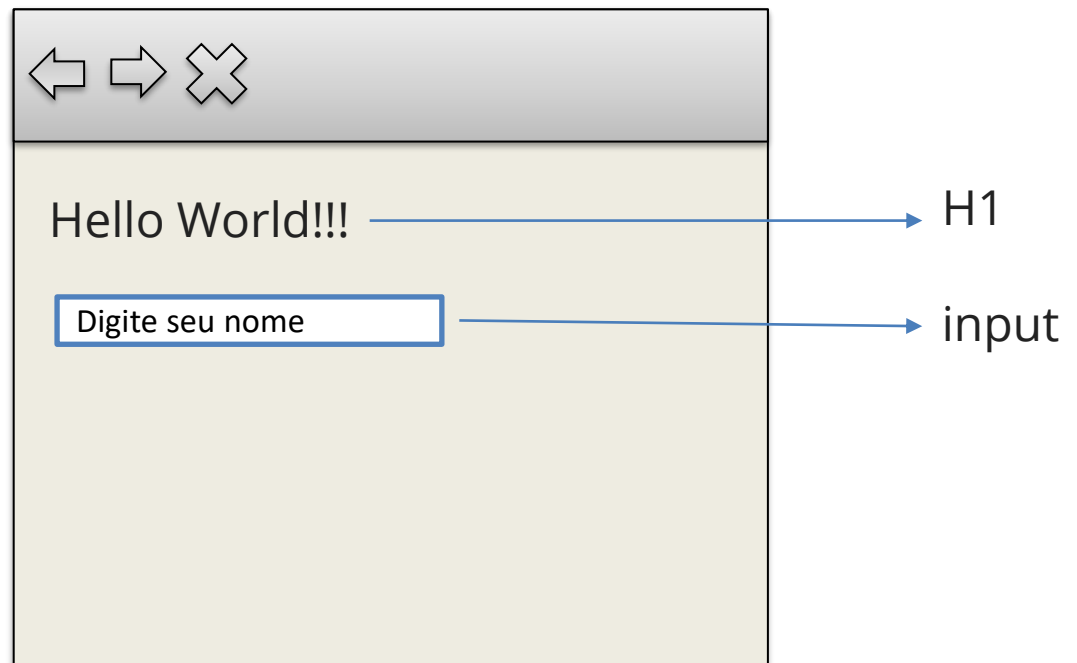
The image shows a code editor interface with two panels. The left panel, titled 'FOLDERS', displays a directory structure: 'JavaScript' (expanded) contains 'Aula 001 - Script no html', which in turn contains 'Aula001\_Script\_no\_Browser.html'. Below this, the file '/\* meu\_script.js' is listed. An orange line originates from this file name and points to the corresponding line in the main editor. The main editor panel shows the content of 'Aula001\_Script\_no\_Browser.html' with line numbers 1 through 21. The code is as follows:

```
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.

The image shows a code editor with an HTML file named `index.html`. The code is as follows:

```
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>
```

Red arrows indicate the flow of execution: one arrow points from the text "Vai gerar um erro interno, veja com F12." to the script block in the code editor, and another points from the same text to the error message in the browser console. A third arrow points from the text "Estou tentando manipular um elemento (id) antes de renderizar o body" to the `id="nome"` attribute in the `<input>` tag.

The browser window shows the page title "Precedência" and the input field with the placeholder text "Insira o seu nome aqui!". The console displays the following error:

```
Uncaught TypeError: Cannot set property 'value' of null
    at index.html:9
```

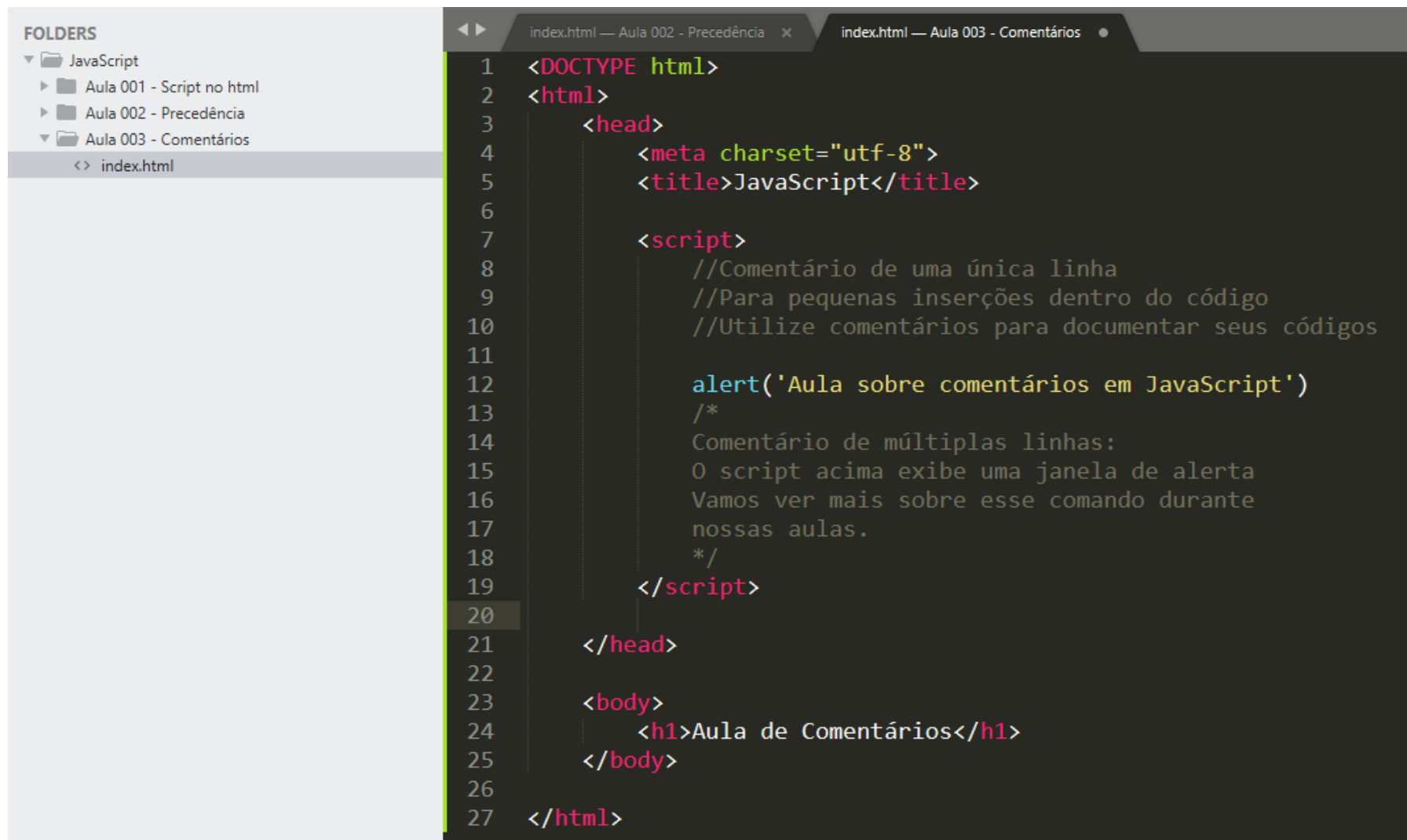
# JavaScript: Ordem de precedência do script

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:

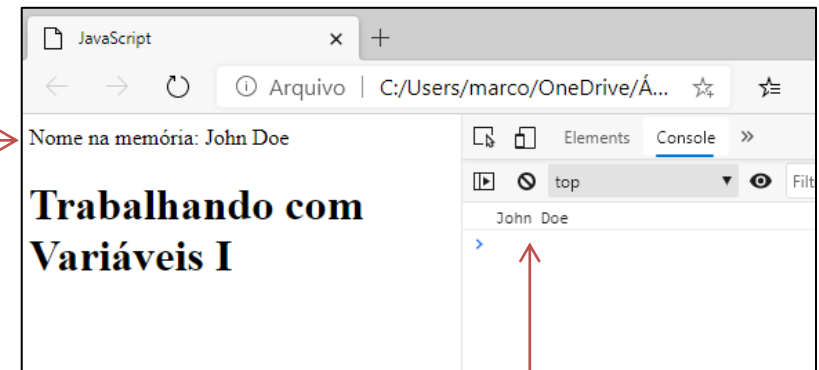
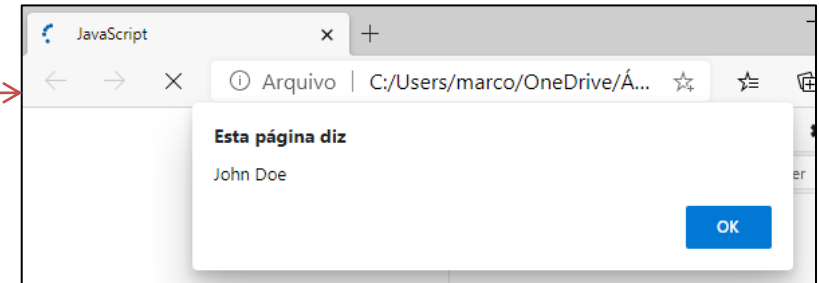
```
let nome é diferente de let NOME
```

# 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

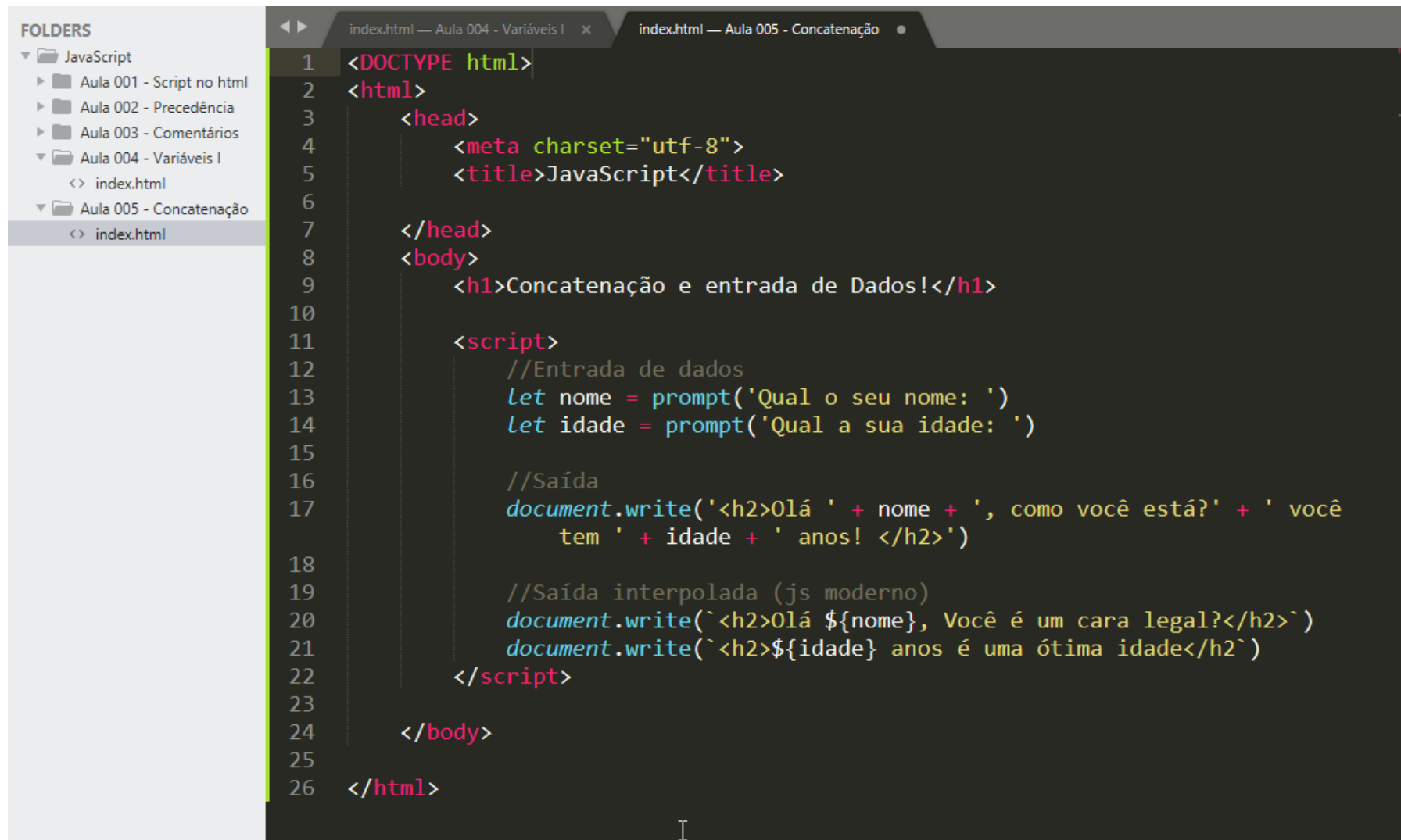
---

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

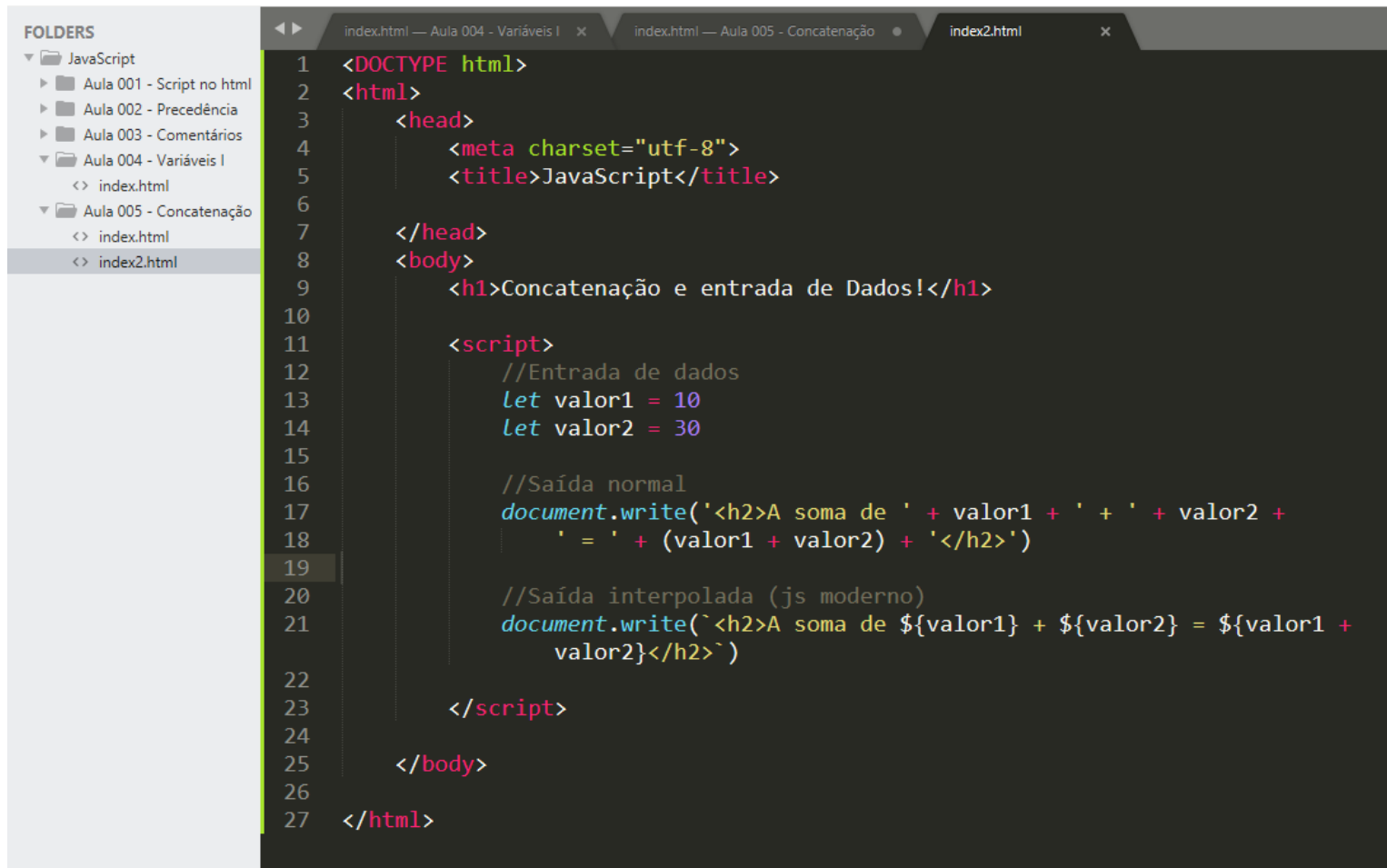


The image shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a folder structure for a JavaScript project, with the current file being 'index.html' in the 'Aula 005 - Concatenação' folder. The code editor displays an HTML document with the following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

```
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 following HTML code:

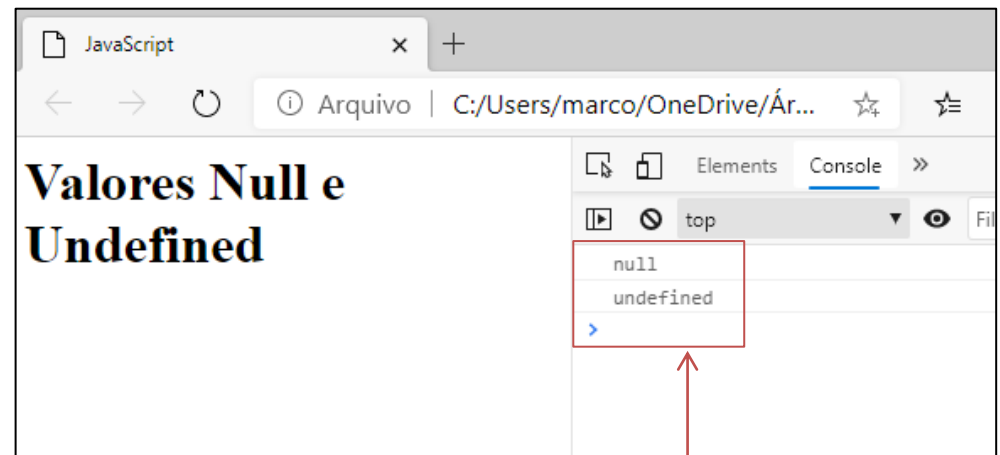
```
1 <DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>JavaScript</title>
6
7   </head>
8   <body>
9     <h1>Concatenação e entrada de Dados!</h1>
10
11    <script>
12      //Entrada de dados
13      let valor1 = 10
14      let valor2 = 30
15
16      //Saída normal
17      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

```
1 <DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>JavaScript</title>
6
7   </head>
8   <body>
9     <h1>Valores Null e Undefined</h1>
10
11    <script>
12      //Null
13      let valor1 = null
14
15      //Undefined
16      let valor2
17
18      //Saída
19      console.log(valor1)
20      console.log(valor2)
21    </script>
22  </body>
23
24 </html>
```



visualização no console F12

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.

```
index.html x
1 <DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>JavaScript</title>
6
7   </head>
8   <body>
9     <h1>Alterando valores das variáveis</h1>
10
11    <script>
12      //Null
13      let valor = null
14
15      //Saída Antes
16      console.log('Antes: ' + valor)
17
18      //Atribuindo um valor
19      valor = 'Valor alterado!'
20
21      //Saída Depois
22      console.log('Depois: ' + valor)
23
24      //Atribuindo outro valor
25      valor = 100.50
26
27      //Saída Depois
28      console.log('Outra alteração: ' + valor)
29
30    </script>
31
32  </body>
33
34 </html>
```



# 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:

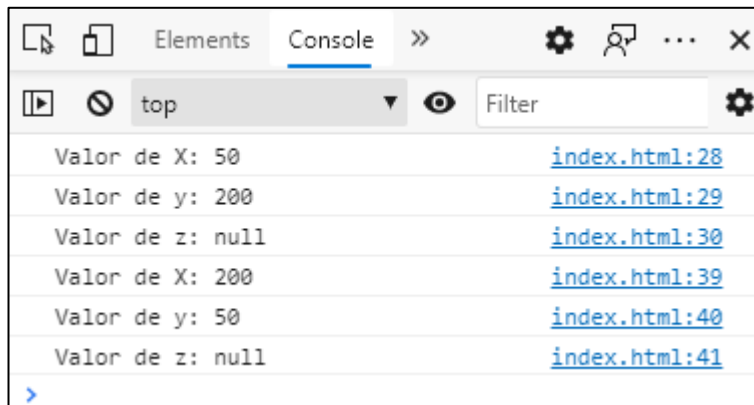
**X = 50**  
**Y = 200**  
**Z = null**

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...



# 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

```
11 <script>
12     //Comando IF ELSE
13     if (Condição) {
14         //Executa Verdadeiro
15     } else {
16         //Executa Falso
17     }
18 </script>
```

Simples

Entenda: um ou outro

```
11 <script>
12     //Comando IF ELSE ENCADEADOS
13     if (Condição) {
14         //Executa Verdadeiro
15     } else if (Condição){
16         //Executa Verdadeiro
17     } else {
18         //Caso as condições anteriores sejam falsas
19     }
20 </script>
```

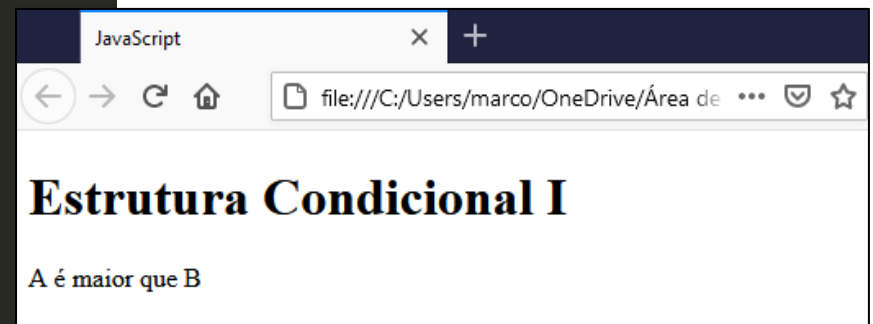
Encadeada

Entenda: várias condições

# JavaScript: Estrutura de Controle 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.

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>JavaScript</title>
6    </head>
7
8    <body>
9      <h1>Estrutura Condicional I</h1>
10
11
12      <script>
13        let a = 30
14        let b = 20
15
16        if (a > b) {
17          document.write('A é maior que B')
18        } else if (b > a){
19          document.write('B é maior que A')
20        } else {
21          document.write('A é igual B')
22        }
23      </script>
24
25    </body>
26  </html>
```



# JavaScript: Estrutura de Controle 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.

Strings são comparadas baseadas na ordenação lexicográfica padrão, usando valores Unicode.

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)

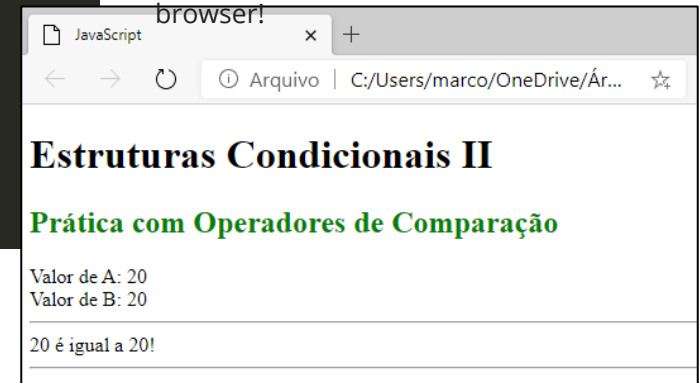
# JavaScript: Estrutura de Controle IF ELSE

Operadores de Comparação praticando...

✓ Igualdade (==)

```
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 = 20
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 igual a ' + b + '!')
25     } else {
26       document.write(a + ' é não é igual a ' + b + '!')
27     }
28     document.write('<hr>')
29   </script>
30 </body>
```

Saída no  
browser!



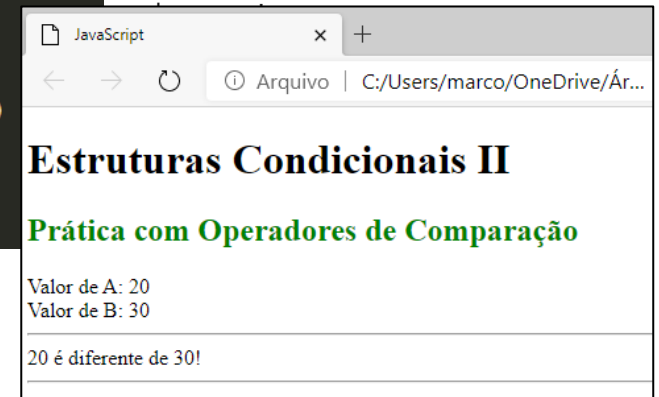
# JavaScript: Estrutura de Controle IF ELSE

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



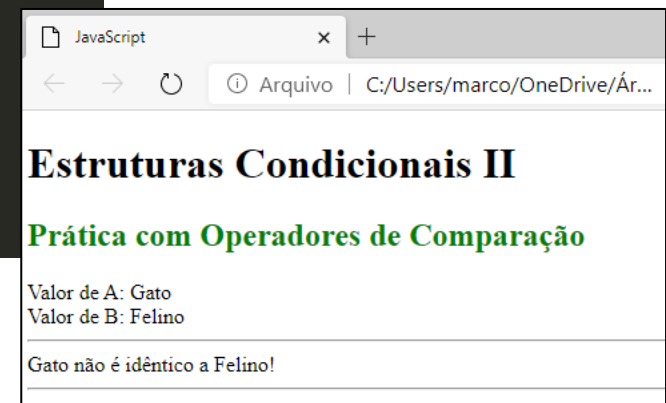
# JavaScript: Estrutura de Controle IF ELSE

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



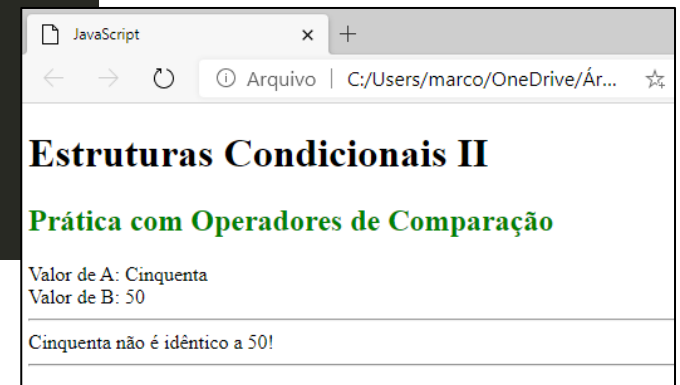
# JavaScript: Estrutura de Controle IF ELSE

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





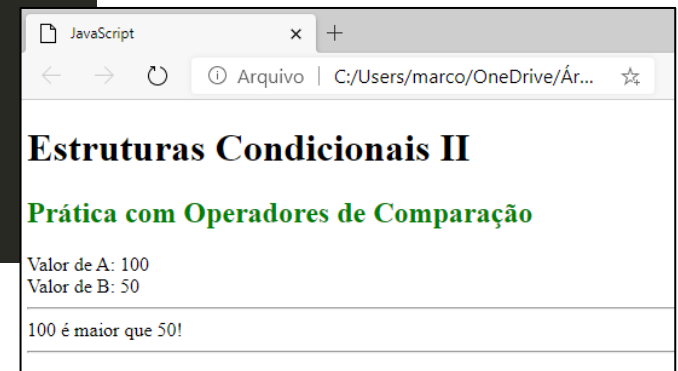
# JavaScript: Estrutura de Controle IF ELSE

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



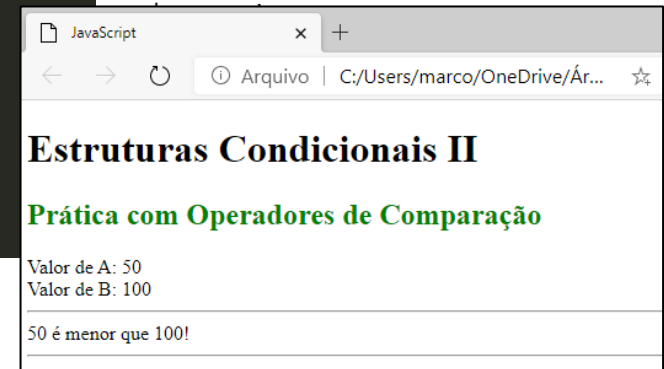
# JavaScript: Estrutura de Controle IF ELSE

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



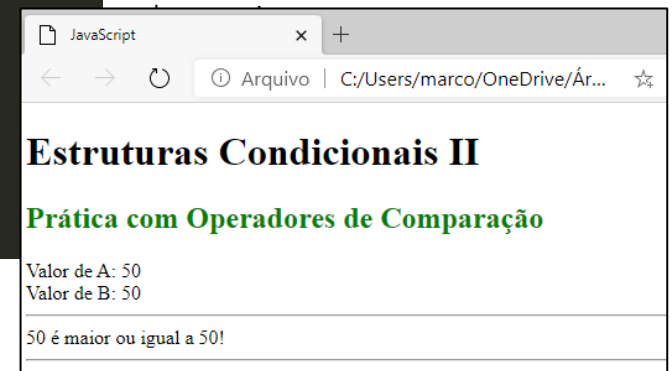
# JavaScript: Estrutura de Controle IF ELSE

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



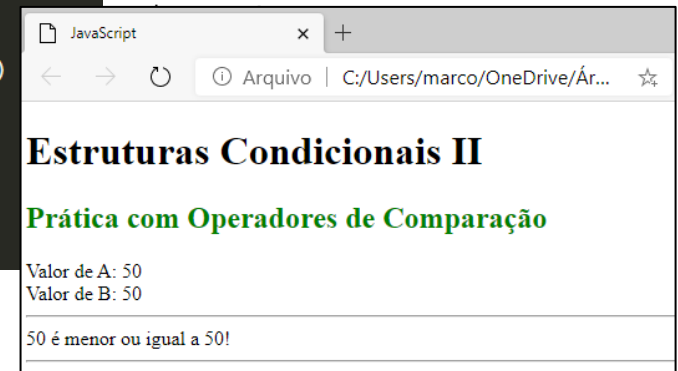
# JavaScript: Estrutura de Controle IF ELSE

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



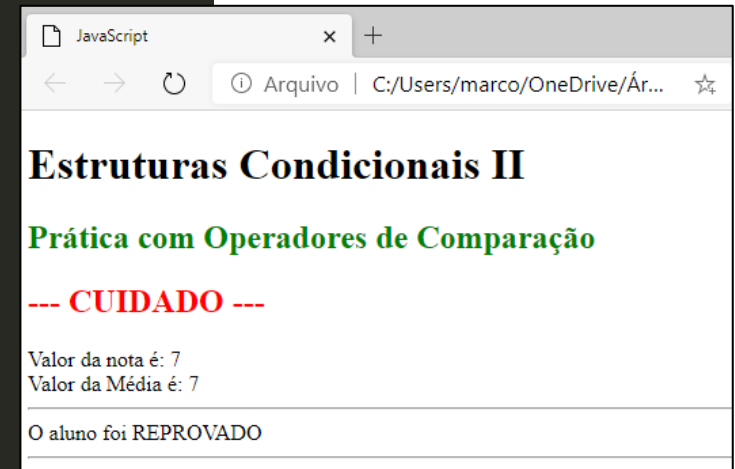
# JavaScript: Estrutura de Controle IF ELSE

## 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   //Condiciona
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!



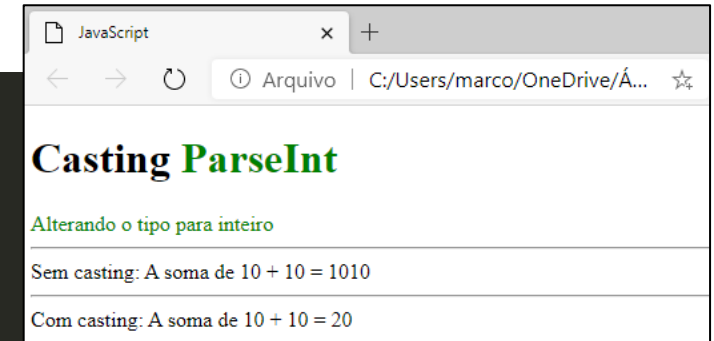
→ Comparação equivocada! Mas funcionaria com operadores simples (=, >, <...)

# 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>
```

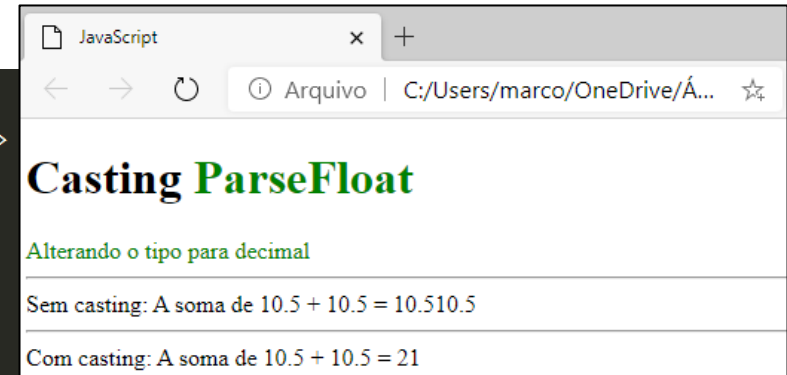


# JavaScript: Casting – Conversão de tipos de dados

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>
```

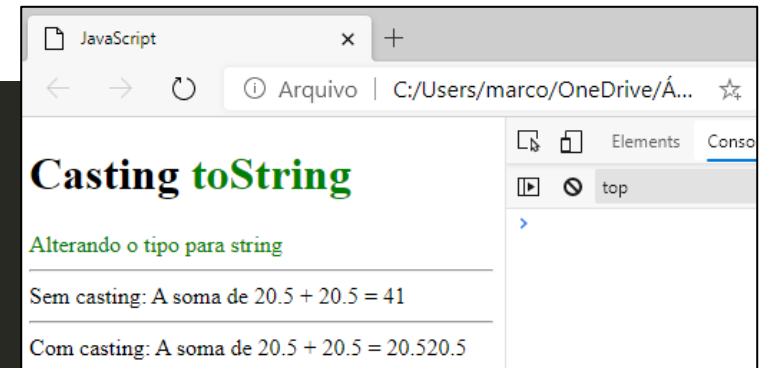


# JavaScript: Casting – Conversão de tipos de dados

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 Castting
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



**Sistema Fecomércio MG, Sesc e Senac**



***Sebastião Marcos***  
*Instrutor de Informática*  
*sebastiao.silva@mg.senac.br*  
*(32) 99196-8445*