

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/>

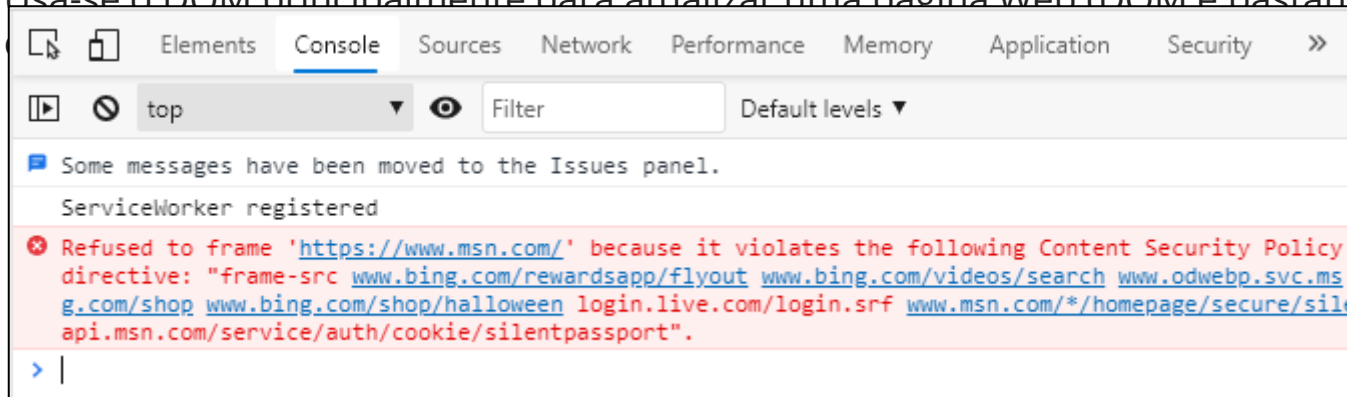
JavaScript

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.

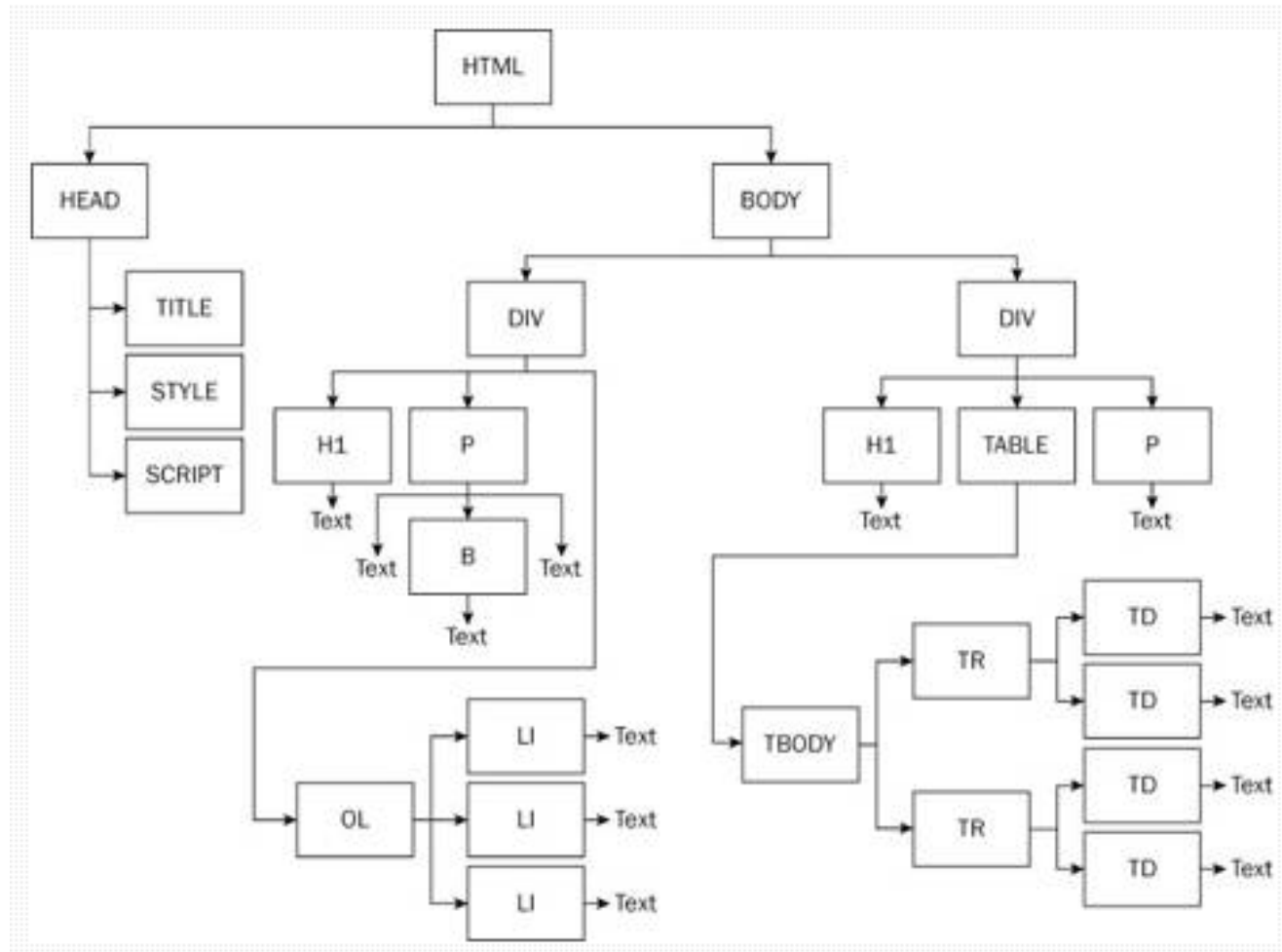
Um dos grandes responsáveis por isso tudo é o **objeto document** que é responsável por conceder ao código Javascript todo o acesso a **árvore DOM do navegador Web**. Portanto, qualquer coisa criado pelo navegador Web no modelo da página Web poderá ser acessado através do objeto Javascript document.

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>`

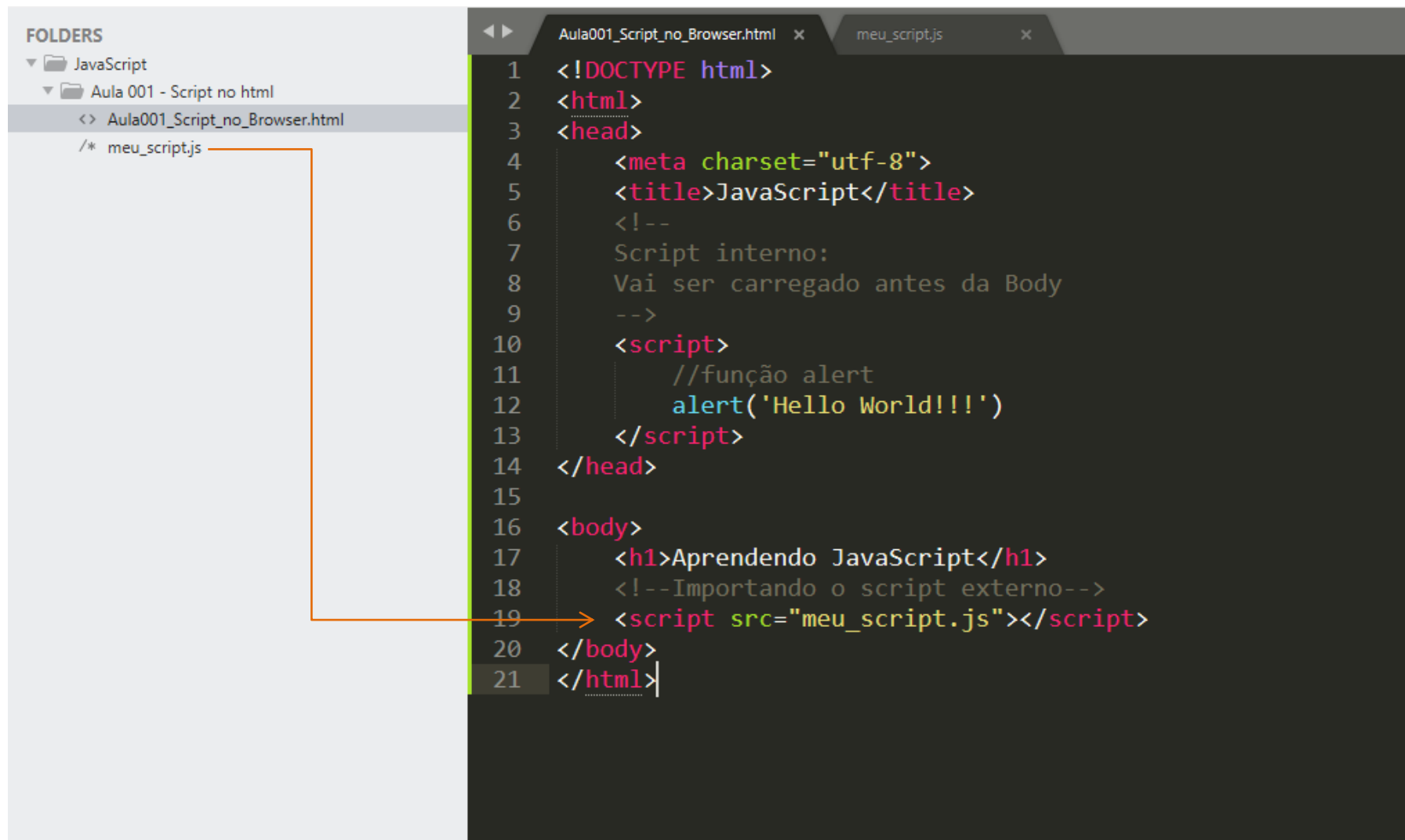
Para essa segunda opção devemos seguir algumas regras:

- 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



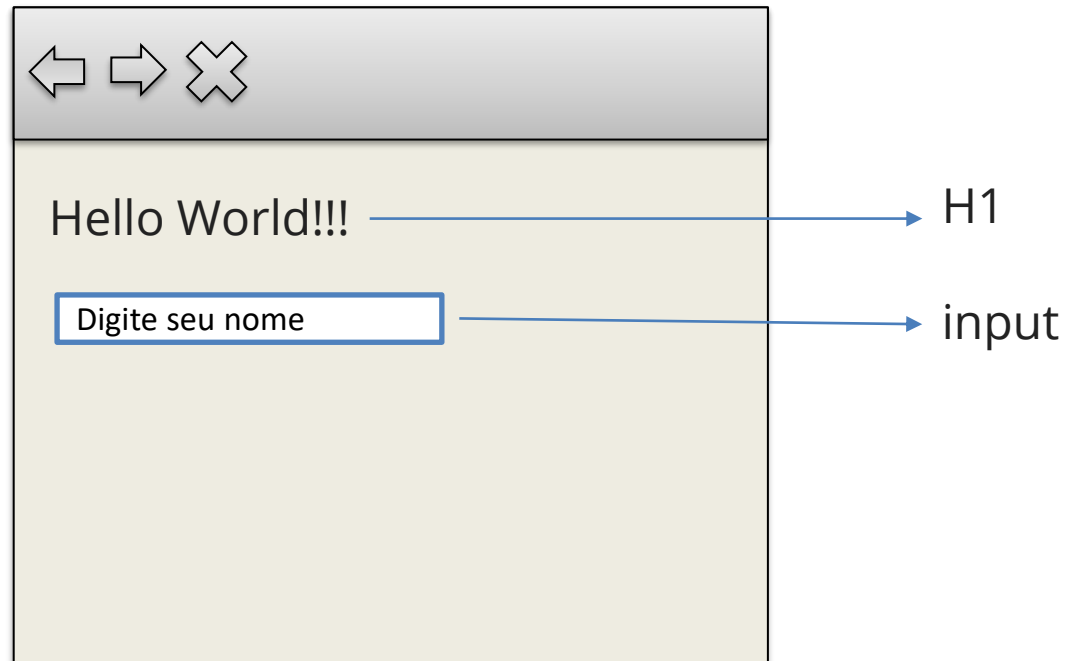
The image shows a code editor interface with two panes. The left pane, titled 'FOLDERS', displays a directory structure: 'JavaScript' (expanded) contains 'Aula 001 - Script no html', which in turn contains 'Aula001_Script_no_Browser.html'. Below this, the file '/* meu_script.js' is listed. An orange line originates from this file name and points to the corresponding line in the main editor pane. The main editor pane shows the HTML code for 'Aula001_Script_no_Browser.html'. The code includes a DOCTYPE declaration, a head section with a meta charset, a title, and a comment about an internal script. The body section contains an h1 heading and a comment about importing an external script. Line 19, which is highlighted with an orange arrow from the file explorer, contains the script tag: `<script src="meu_script.js"></script>`. The file explorer also shows a file named 'meu_script.js'.

```
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: from the text "Vai gerar um erro interno, veja com F12." to the script block in the code editor, then to the browser console, and finally to the text "Estou tentando manipular um elemento (id) antes de renderizar o body".

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

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

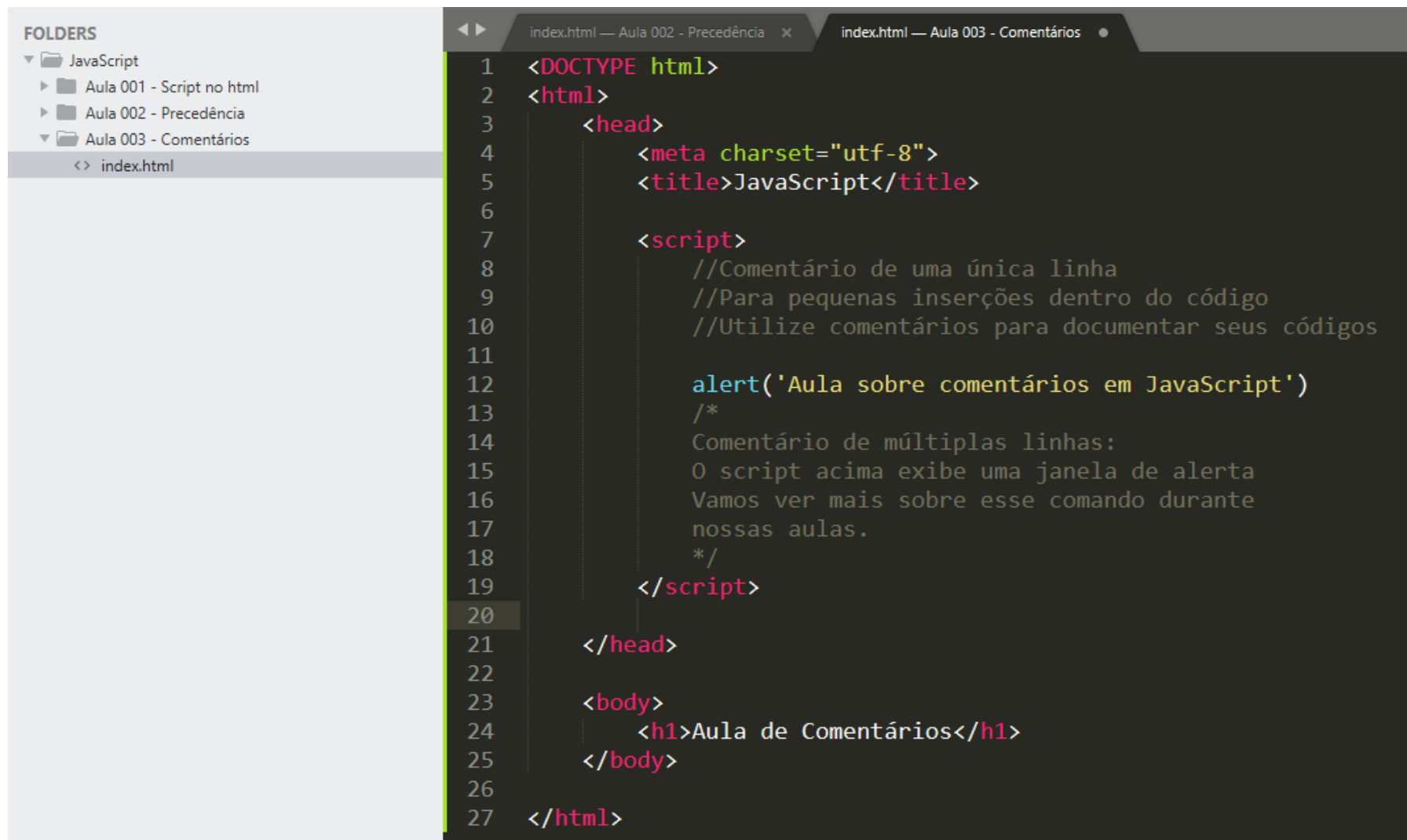
JavaScript: Ordem de precedência do script

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:

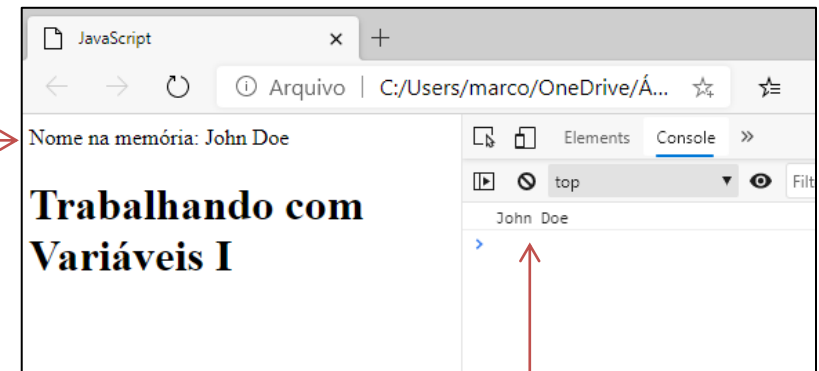
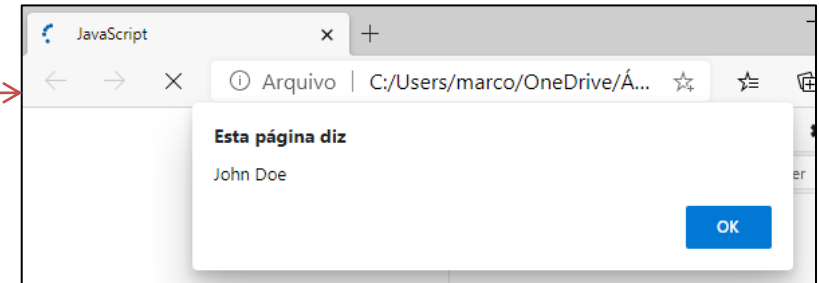
```
let nome é diferente de let NOME
```

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>
```



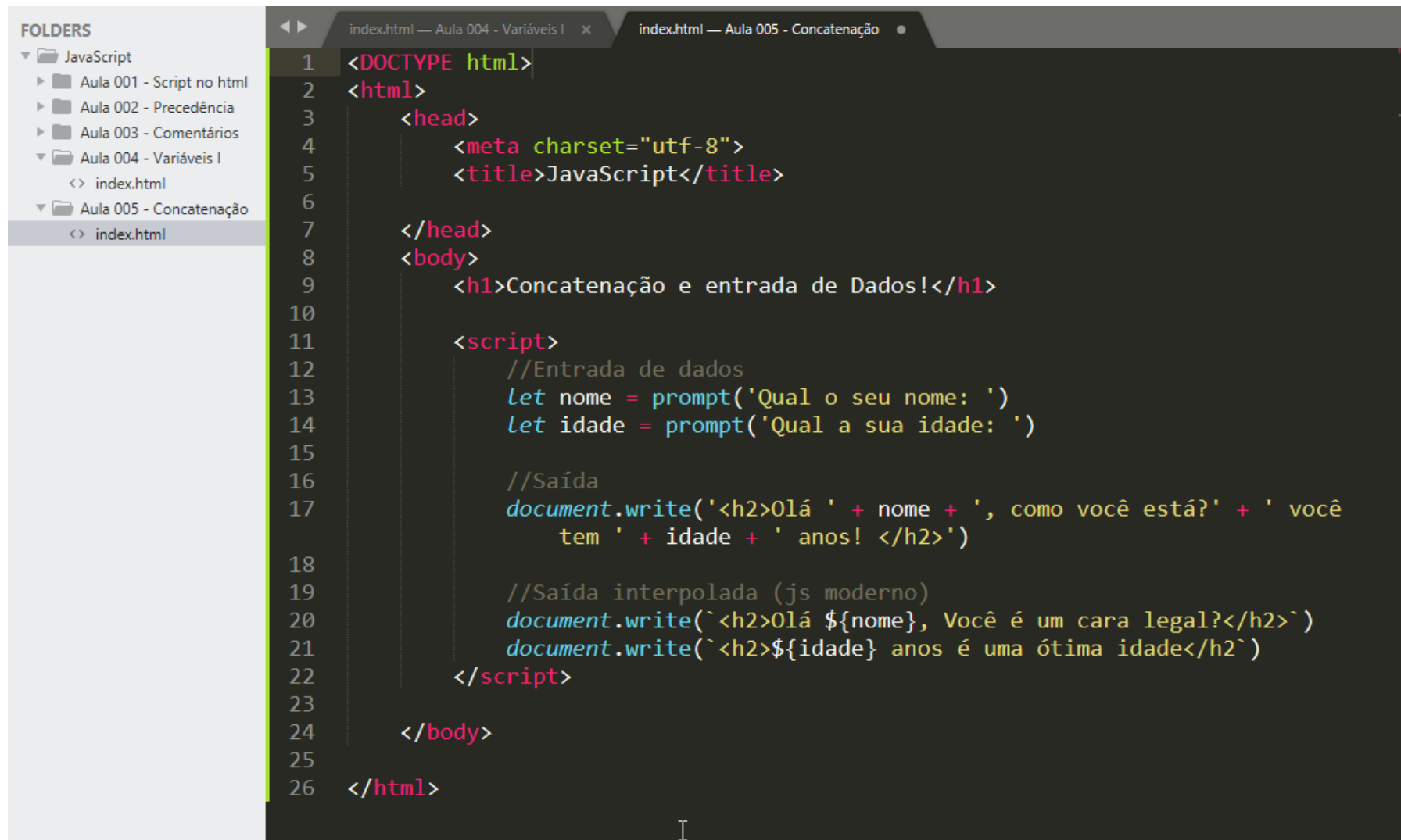
JavaScript: Concatenação e Entrada de dados

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

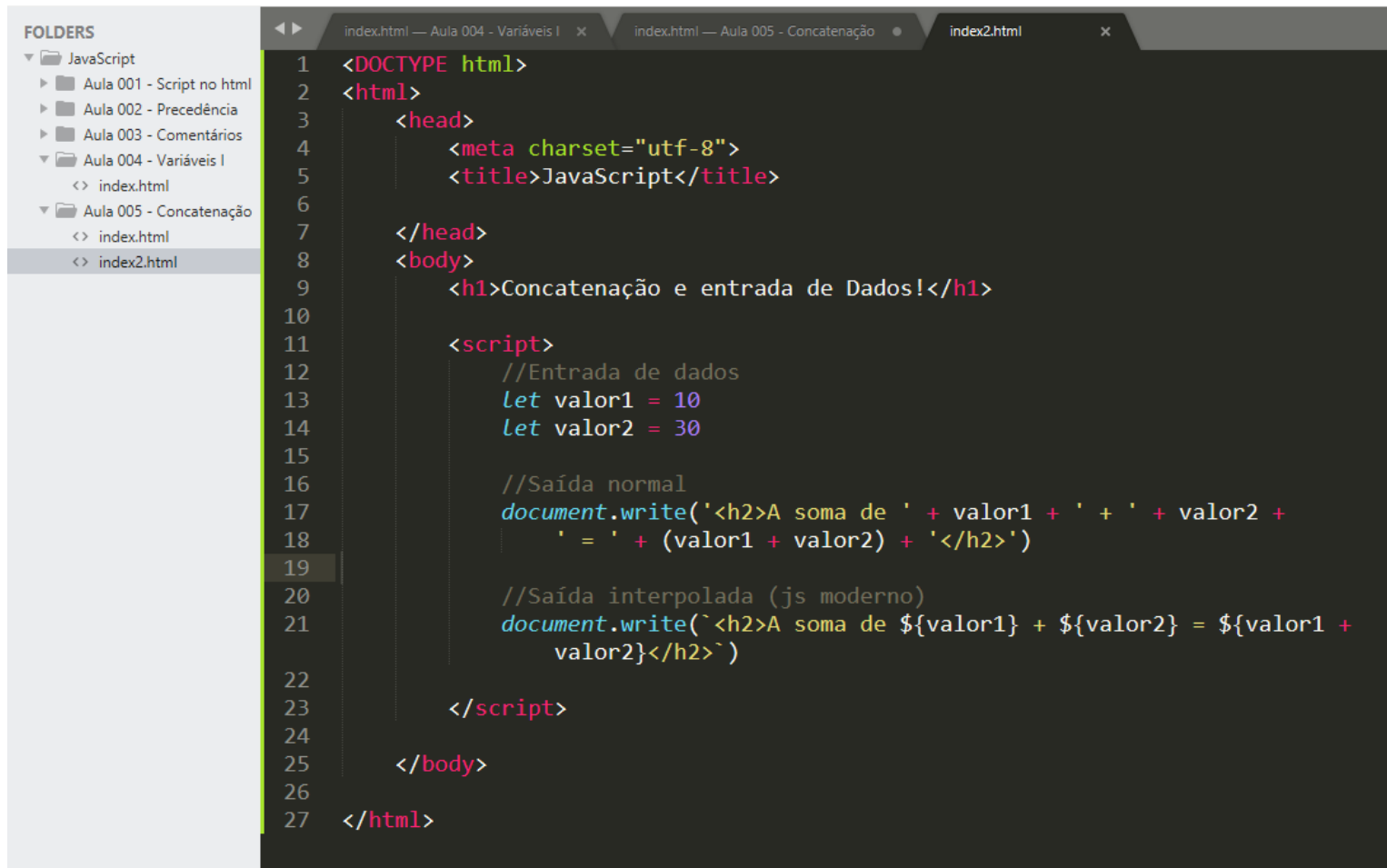


The image shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a folder structure for a JavaScript project, with the current file being 'index.html' in the 'Aula 005 - Concatenação' folder. The code editor displays an HTML document with the following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

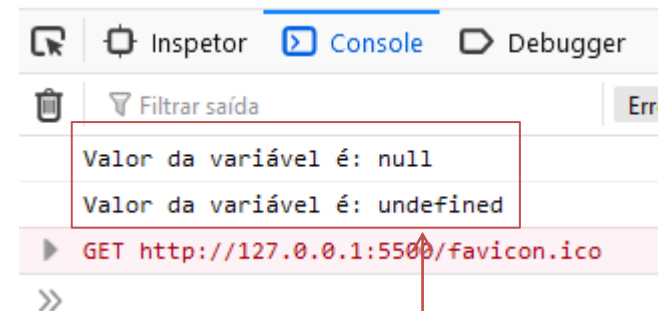
```
1 <DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>JavaScript</title>
6
7   </head>
8   <body>
9     <h1>Concatenação e entrada de Dados!</h1>
10
11    <script>
12      //Entrada de dados
13      let valor1 = 10
14      let valor2 = 30
15
16      //Saída normal
17    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

```
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>
```



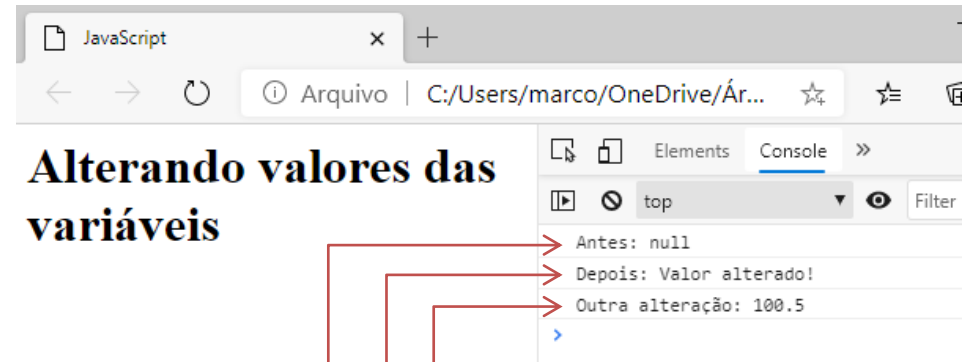
É importante que os valores de Null e Undefined sejam tratados durante a lógica de programação

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

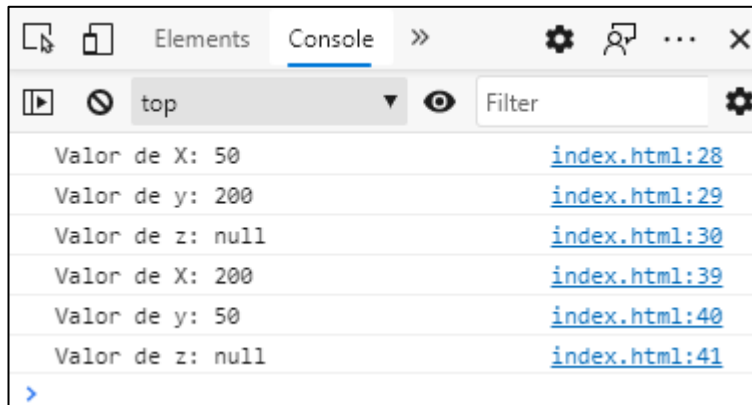
JavaScript: Desafio

Crie um programa que altere o valor das variáveis X e Y baseados nos seguintes dados:

X = 50
Y = 200
Z = null

O desafio é inverter o valor 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!

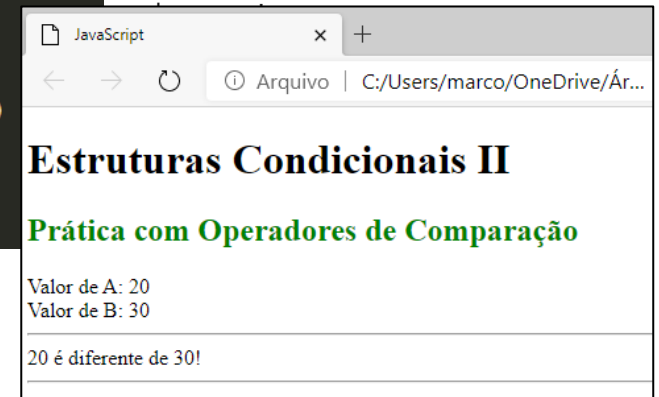
JavaScript: Estrutura de Controle IF ELSE

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



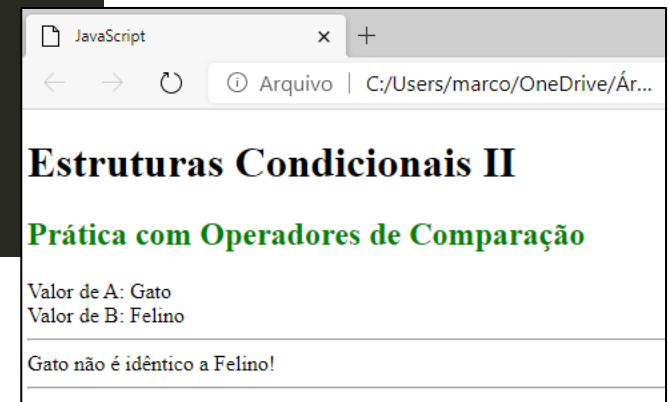
JavaScript: Estrutura de Controle IF ELSE

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



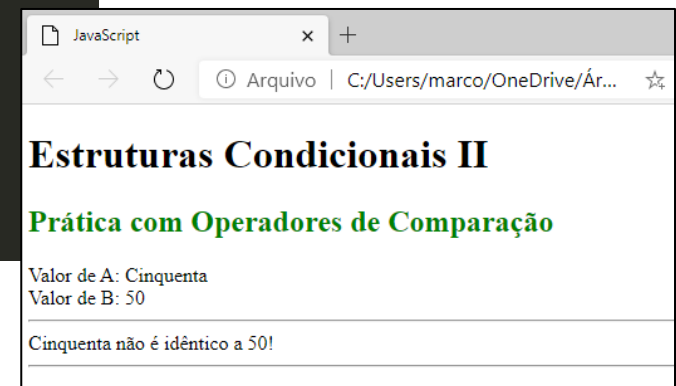
JavaScript: Estrutura de Controle IF ELSE

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



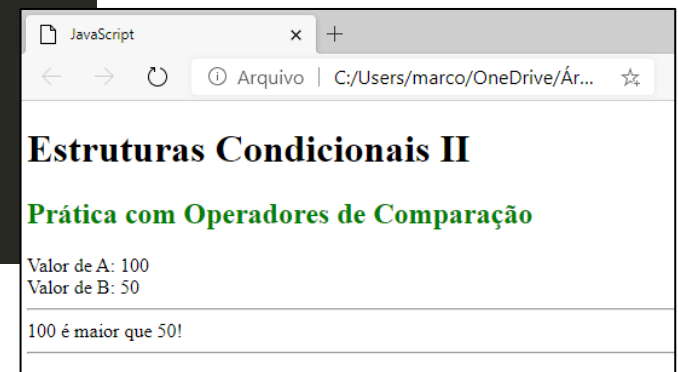
JavaScript: Estrutura de Controle IF ELSE

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



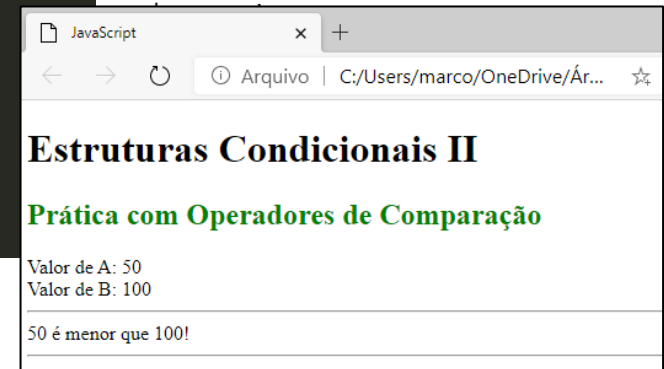
JavaScript: Estrutura de Controle IF ELSE

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



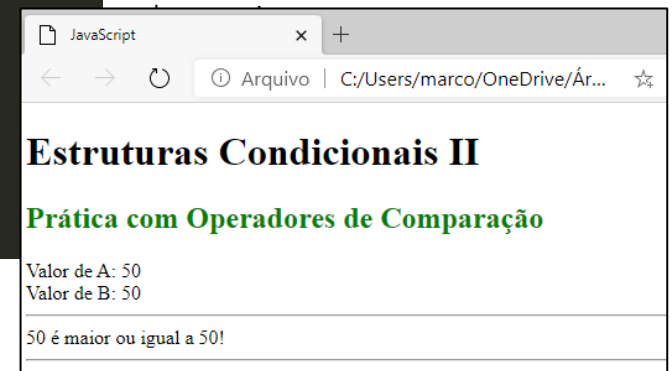
JavaScript: Estrutura de Controle IF ELSE

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



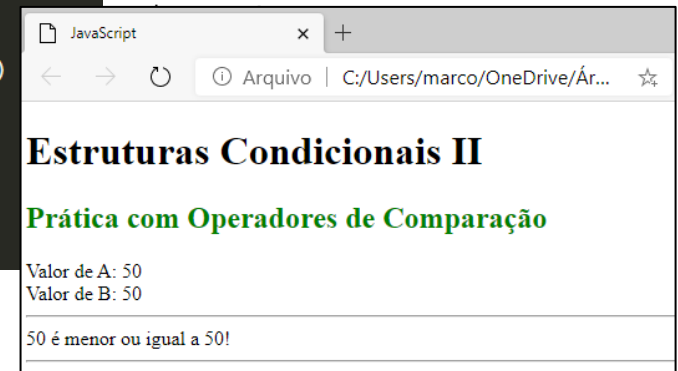
JavaScript: Estrutura de Controle IF ELSE

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



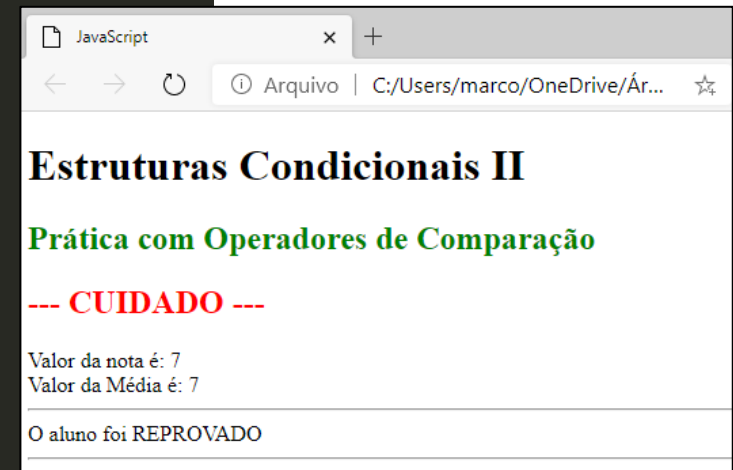
JavaScript: Estrutura de Controle IF ELSE

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   //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!



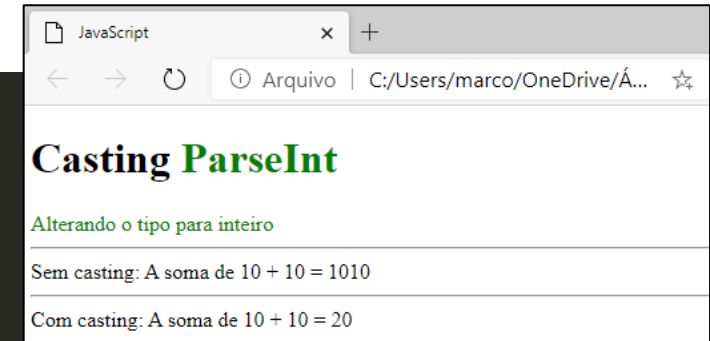
→ Comparação equivocada! Mas funcionaria com operadores simples (=, >, <...)

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}`)
```

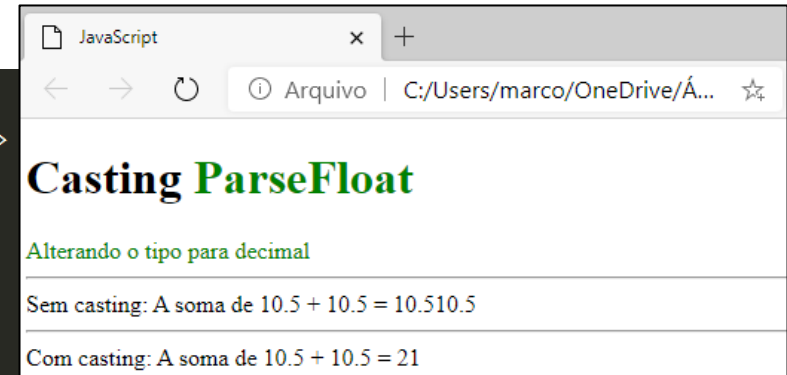


JavaScript: Casting – Conversão de tipos de dados

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>
38 </body>
```

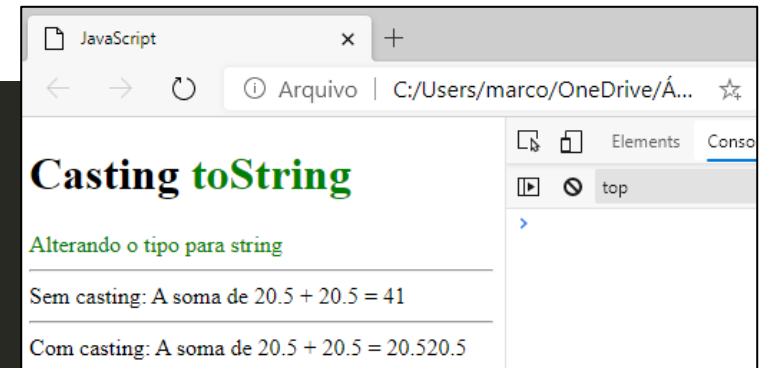


JavaScript: Casting – Conversão de tipos de dados

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.**

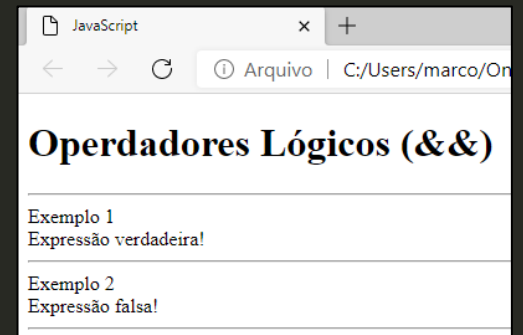
Os operadores lógicos são:

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

JavaScript: Operadores lógicos

Testando o && verdadeiro e Falso

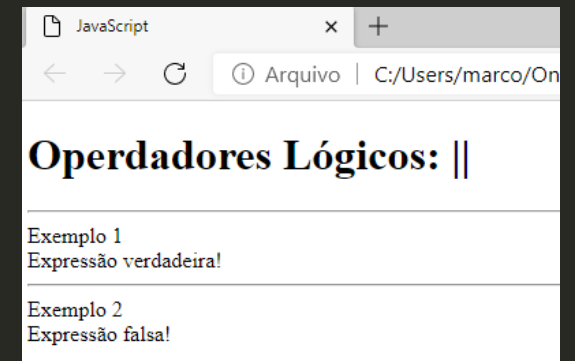
```
8      <body>
9          <h1>Operadores Lógicos (&&)</h1>
10         <hr>
11
12         <script>
13             //Operador &&: Verdadeiro
14             document.write("Exemplo 1 <br>")
15             if (10 == 10 && 20 < 40 && 'a' != 'b') {
16                 document.write("Expressão verdadeira!")
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

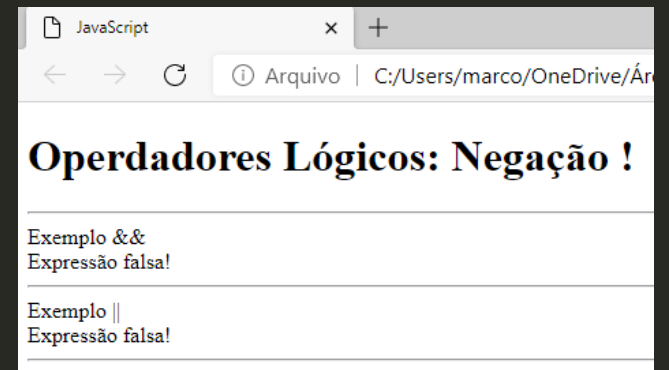
```
8   <body>
9     <h1>Operadores Lógicos: || </h1>
10    <hr>
11
12    <script>
13
14      //Operador || (OU): Verdadeiro
15      document.write("Exemplo 1 <br>")
16      if (10 == 10 || 20 < 40 || 'a' == 'b') {
17        document.write("Expressão verdadeira!")
18      } else {
19        document.write("Expressão 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>
```

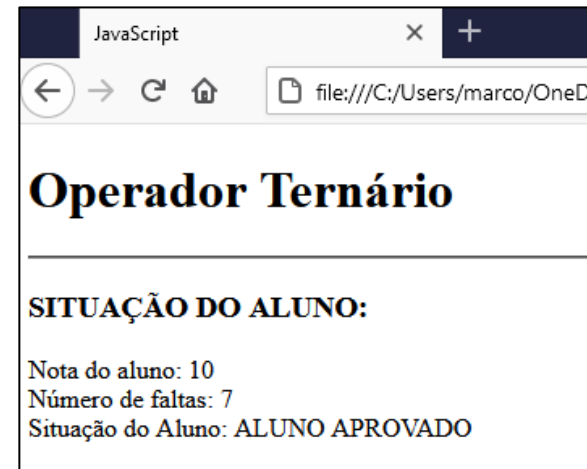


JavaScript: Operador Ternário

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 **Multiplicação** ou **Produto (*)**: Produto entre valores

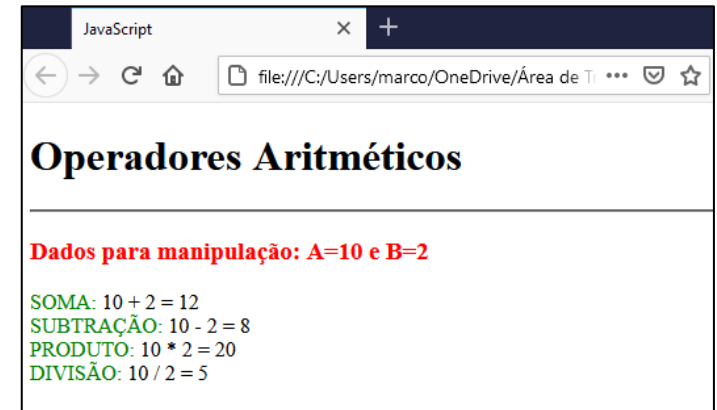
Operador de **Divisão (/)**: Quociente entre valores

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-->
72
73
```

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 número!

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 60m²