



**CENTRO UNIVERSITÁRIO SENAC - SANTO AMARO**  
**ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**  
**PROGRAMAÇÃO WEB**

**ALUNO: LEONARDO RAVANELLI COELHO**

**ORIENTADOR: PROF. CARLOS HENRIQUE VERISSIMO PEREIRA**

**A RELAÇÃO ENTRE DOCUMENT OBJECT MODEL E O JAVASCRIPT**

**SÃO PAULO**

**2022**

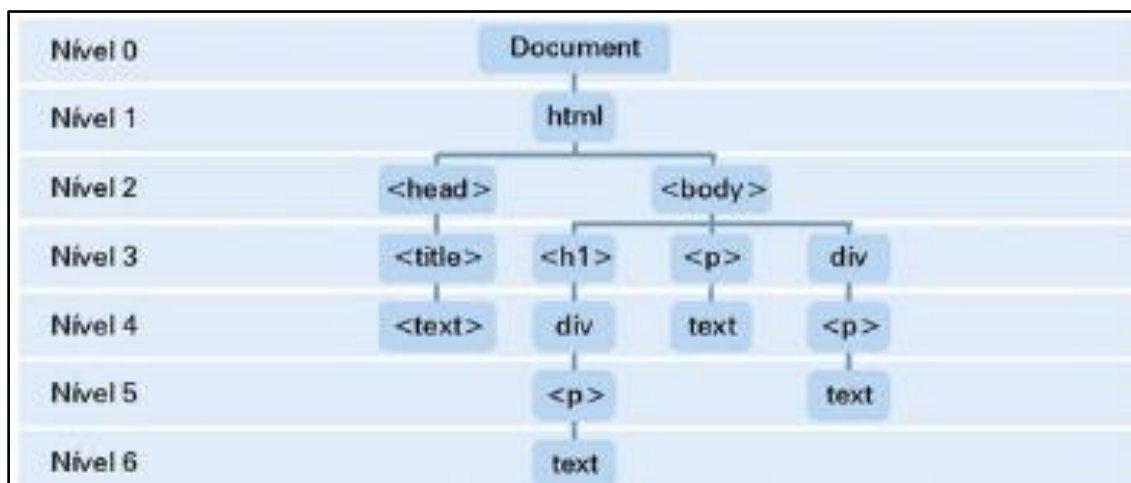
## Sumário

1.	Capítulo .....	1
1.1.	OQUE É O DOM? .....	1
1.1.1.	NIVEL 0 .....	1
1.1.2.	NIVEL 2 .....	2
1.1.3.	NIVEL 1 .....	2
1.2.	RELAÇÃO ENTRE DOM E JAVASCRIPT .....	3
1.3.	ELEMENTOS DA IMPLEMENTAÇÃO DO JAVASCRIPT/DOM .....	4
1.3.1.	getElementById .....	4
1.3.2.	getElementsByTagName .....	5
1.3.3.	getElementsByClassName .....	5
1.3.4.	querySelector .....	5
2.	Capítulo .....	6
3.	Capítulo .....	7
3.1.	BIBLIOGRAFIA .....	7

# 1. Capítulo

## 1.1. OQUE É O DOM?

O Document Object Model (DOM) é uma interface de desenvolvimento web que expõe todos os objetos em uma página HTML, ele é fundamental para representar e manipular o conteúdo de documentos HTML e XML. A API A modificação desses objetos se torna possível graças a sua montagem em forma de árvore, cada objeto possui métodos e atributos que viabilizam os recursos que são capazes de alterar essa estrutura dinamicamente, nele também pode haver marcações ou elementos HTML, como `<body>` e `<p>` nós representando strings de texto. Um documento HTML também pode conter nós representando comentários HTML.



O DOM existe desde o tempo em que os navegadores começaram a suportar a linguagem JavaScript, em uma época que os desenvolvedores queriam acessar bits de HTML e alterar suas propriedades. A principal função do DOM é permitir o acesso a esses elementos HTML da página web. Desde o seu surgimento existem três níveis de modelo de documentos por objeto, que são:

### 1.1.1. NÍVEL 0

suportado pelo Netscape 2, surgiu no mesmo período do JavaScript. Por razões de compatibilidade com versões anteriores, os navegadores mais avançados

ainda oferecem suporte a esse nível, fazendo com que a microsoft copie o Netscape DOM para o Internet Explore 3.

#### **1.1.2. NIVEL 2**

suportado pelo Netscape 4 e pelo Internet explorer 4 e 5 esse nível se tornou obsoleto.

#### **1.1.3. NIVEL 1**

chamado de nível 1 ou DOM W3C, tem compatibilidade com mozilla e Internet Explorer 5, ele não fornece apenas um modelo exato para todo o documento HTML, mas também permite alterar um documento, retirar um paragrafo alterar o layout de uma tabela.

## 1.2. RELAÇÃO ENTRE DOM E JAVASCRIPT

A plataforma DOM é importante e intuitiva para manipular dinamicamente as informações de uma página HTML em tempo de execução com uma linguagem Javascript.

Quando você abre uma página web em seu navegador, ele resgata o texto em HTML da página e o interpreta. O navegador constrói um modelo da estrutura do documento e depois usa esse modelo para desenhar a página na tela. Um dos "brinquedos" que um programa em JavaScript possui disponível em sua caixa de ferramentas é essa representação do documento. Você pode lê-la e também alterá-la. Essa representação age como uma estrutura viva de dados: quando modificada, a página na tela é atualizada para refletir as mudanças.

Então podemos ver que Sem o DOM, o Javascript não teria um modelo para representar seus objetos. Afinal, ela não teria uma noção da página web e de seus componentes.

Assim, enquanto o DOM representa a estrutura e os componentes da página, o JavaScript acessa e manipula esses conteúdos.

### 1.3. ELEMENTOS DA IMPLEMENTAÇÃO DO JAVASCRIPT/DOM

A maioria dos programas JavaScript do lado do cliente funciona manipulando de alguma forma um ou mais elementos de documento. Quando esses programas começam, podem utilizar a variável global `document` para se referirem ao objeto `Document`. Contudo, para manipular elementos do documento, eles precisam de algum modo obter ou selecionar os objetos `Element` que se referem a esses elementos de documento. O DOM define várias maneiras de selecionar elementos em HTML, são eles `getElementById`; `getElementsByTagName`; `getElementsByClassName`.

#### 1.3.1. `getElementById`

viabiliza a localização do elemento pelo atributo `id` criado no código HTML  
exemplo:

```
1  <p id="id_parag" class="nome_classe"> Centro  
    Universitário </p>  
2  var wp = document.getElementById('id_parag');  
3  wp.style.backgroundColor = 'blue'
```

O exemplo anterior apresenta, na linha 1, um exemplo de linha escrita em HTML, cuja tag de parágrafo `<p>` apresenta o parâmetro `id`, que o identifica como `id_parag`. Na linha 2, temos um exemplo de linha em JavaScript usando o método `getElementById`, que faz referência ao `id` da linha 1. Na terceira linha, atribuímos à variável `wp` outra cor de fundo do texto que está no parágrafo identificado pelo `id`.

#### 1.3.2. `getElementsByTagName`

também podemos localizar o elemento HTML pelo nome da tag, como `p` para parágrafos, `div` para divisões, `h1` para cabeçalhos e muitos outros.  
exemplo:

```
var parag = document.getElementsByTagName('p');
```

Note que está escrito `elements`, no plural, pois pode existir mais de um elemento pela tag `name`, o que possibilita que os elementos sejam gravados em um array, ao qual podemos nos referenciar, como no próximo exemplo, utilizando o índice do vetor.

### 1.3.3. `getElementsByClassName`

assim como o `getElementsByTagName`, retorna um array de elementos.

### 1.3.4. `querySelector`

Outra forma de selecionar elementos é utilizando o `querySelector`, que retoma o valor do primeiro elemento dentro do documento em HTML que seja idêntico ao grupo ou conjunto de seletores.

```
1  <html>
2      <head>
3          <title> Interface para entrada de dados </
title>
4      </head>
5      <body>
6          <input type="text"  class="textocss">
7          <button onclick="recebeentrada()">Receber</
button>
8
9          <script>
10             function recebeentrada() {
11                 var textorecebido = document.
querySelector('.textocss').value;
12                 var recebefilho = textorecebido.
childNodes;
13                 alert(textorecebido+recebefilho);
14             }
15         </script>
16     </body>
17 </html>
```

Na linha 11, dentro da função, o `document.querySelector` procura no código HTML a primeira ocorrência da classe `"textocss"`. Ao procurar uma classe, sempre se deve inserir o ponto antes do nome.

## 2. Capítulo

```
9  <!DOCTYPE html>
10 <html lang="pt-br">
11 <head>
12   <meta charset="UTF-8">
13   <meta name="viewport" content="width=device-width, initial-scale=1.0">
14   <title>Números com JS</title>
15   <style>
16     body { font: 12pt Arial; }
17     button { font-size: 12pt; padding: 30px; }
18   </style>
19 </head>
20 <body>
21   <h1>Senac - TADS - PW - 2º Semestre </h1>
22   <h2>Aula #02 - Introdução ao JS</h2>
23   <h3>Fazendo Cálculos</h3>
24   <button onclick="calculadora()">Clique para calcular</button>
25   <section id="res">
26     <p>Atenção: O resultado será colocado aqui...</p>
27   </section>
28
29   <script>
30     // O que fica aqui fora vai executar automaticamente, SEMPRE que o site for carregado
31     window.alert('Seja bem-vindo(a) ao meu site!')
32     // Já a função calcular() só vai executar quando o usuário pressionar o botão
33     function calculadora() {
34       let n1 = Number(window.prompt('Digite um número: '))
35       let res = document.querySelector('section#res')
36
37       res.innerHTML = `<p>O dobro de ${n1} é ${n1*2} e a metade é ${n1/2}!</p>`
38     }
39   </script>
40 </body>
41 </html>
```

- A implementação começa na linha 29 quando abre o `<script>`.
- Na linha 31, é puxado dos elementos da árvore do DOM, o `window.alert` que usando o Javascript eo DOM, envia uma mensagem em janela para o Usuário.
- Na linha 34 é utilizado o `window.prompt` que é mais uma interação do DOM eo JavaScript, com isso é aberta uma janela, que pede informações ao usuário.
- Na linha 35 Você está acessando o section que tem a id que é representado por um # de res.
- Na linha 37 você está pegando o res com todas as tags nele e transformando na mensagem que esta depois do =.



## 3. Capítulo

### 3.1. BIBLIOGRAFIA

MAIA, Rômulo. **Linguagens de script para web**. 01. ed. São Paulo: Editora Senac São Paulo 2021.

JUNIOR, Luis. **JavaScript – interatividade para web**. 01 .ed. São Paulo: Editora Senac São Paulo.

FLANAGAN, David. **JavaScript: O Guia Definitivo**. 06 ed. Porto Alegre: Bookman.

HAVERBEKE, Marijin. **Eloquent JavaScript: A Modern Introduction to Programming**. 02. ed. São Francisco: No Starch Press 2014.