6. MANIPULAÇÃO DO DOM

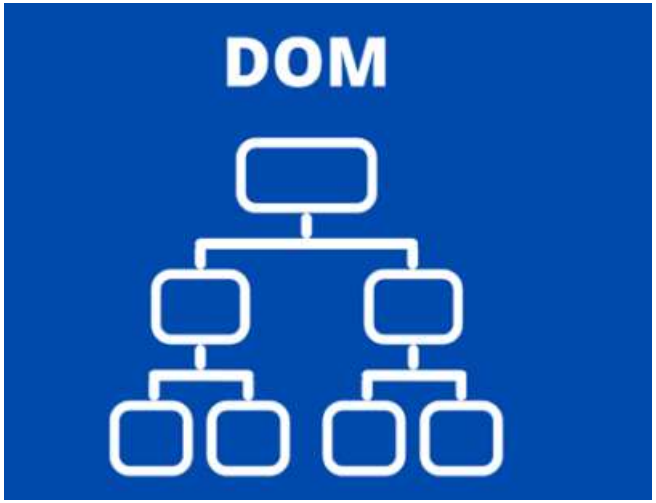
INTRODUÇÃO AO DOM

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.

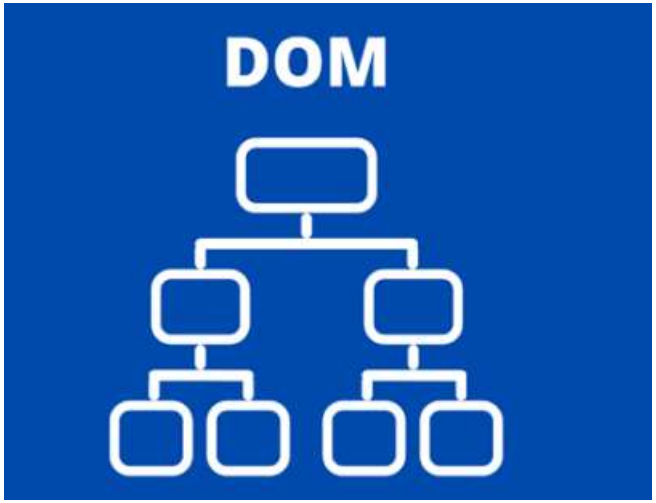
O QUE É O DOM?

O **Document Object Model** (DOM) é uma interface de programação para os documentos HTML e XML. Representa a página de forma que os programas possam alterar a estrutura do documento, alterar o estilo e conteúdo. O DOM representa o documento com nós e objetos, dessa forma, as linguagens de programação podem se conectar à página.



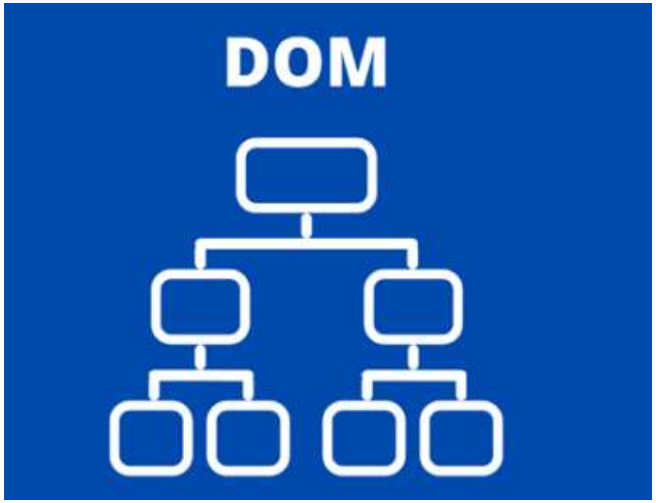
O QUE É O DOM?

Usa-se o DOM principalmente para atualizar uma página Web (DOM é bastante utilizado com Ajax) ou quando se quer construir uma interface de usuário avançada. Com o DOM pode-se mover itens dentro de uma página ou criar efeitos CSS bastante interessantes sem precisar nem mesmo recarregar a página.



POR QUE O DOM É IMPORTANTE?

- Deixa o site **dinâmico** e **interativo**;
- Permite criar efeitos visuais, menus, sliders, modais, etc.;
- Torna possível validar formulários antes de enviar;
- É a base para usar bibliotecas como **React**, **Vue**, e **Angular**.



OBJETO DOCUMENT

Através do objeto document pode-se ter acesso a um grande número de propriedades. Segue abaixo algumas propriedades que podem ser utilizadas com o objeto document:

Propriedade	Descrição
`documentElement`	Captura o elemento raiz <html> de um documento HTML.
`getElementById`	Busca um elemento da página Web com o uso do atributo id do elemento.
`createElement`	Cria um nodo elemento na página.
`createAttribute`	Cria um nodo atributo na página.
`createTextNode`	Cria um nodo texto na página.
`getElementsByTagName`	Retorna um array dos elementos com o mesmo nome.
`appendChild`	Insere um novo elemento filho.
`removeChild`	Remove um elemento filho.
`parentNode`	Retorna o nodo pai de um nodo.

OBJETO DOCUMENT

Segue abaixo um exemplo de uma árvore DOM de uma página Web. Pode-se notar que todos os elementos da página Web estão disponíveis para serem manipulados.

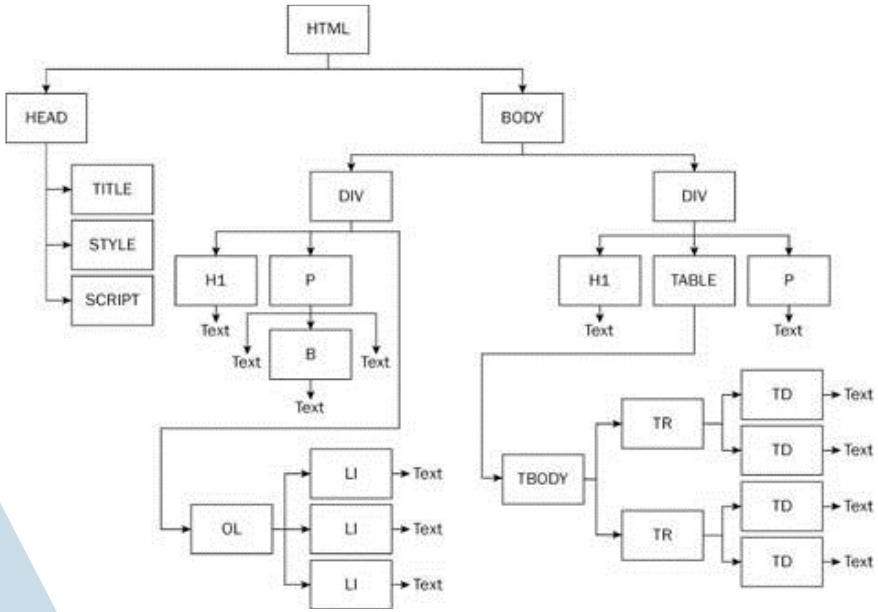


Figura 1. Exemplo de uma árvore DOM de uma página Web

OBJETO DOCUMENT

Todas as páginas Web são de alguma forma uma árvore. Isso se deve ao fato de podermos ver uma página Web como uma árvore, com uma raiz como o elemento HTML e os seus filhos como o HEAD e o BODY que por sua também possuem elementos filhos e assim sucessivamente.

Os elementos que não possuem filhos são chamados de nós folhas, como por exemplo os elementos TITLE, STYLE, SCRIPT, LI, H1, P, TD demonstrados acima. Note que Text é um texto que está dentro de um elemento. O nó <TD> por exemplo também é considerado um nó, mas um nó de tipo diferente (tipo texto).

OBJETO DOCUMENT

Essa estrutura de árvore é a forma que o navegador organiza as marcações do HTML, é dessa forma que o navegador Web enxerga um documento HTML. A leitura da árvore se dá sempre da esquerda para a direita, assim teremos a página Web original.

Uma boa prática para evitar que os navegadores apresentem a pagina Web de formas diferentes é sempre escrever HTML padrão, se possível sempre validando a página HTML. Fechar os elementos corretamente, utilizar tags atuais evitando as tags desatualizadas ajudam os navegadores a exibirem uma página Web de maneira correta.

PERCORRENDO ÁRVORES DOM

Como visto anteriormente tem-se uma árvore na qual podemos percorrer-la utilizando a variável document no código Javascript. Como cada nó tem um pai e a maioria tem um nó filho, pode-se percorrer essa árvores DOM subindo e descendo nela usando essas conexões. Lembrando que tudo que há na árvore DOM é um nó e nisso incluímos também os textos, elementos, atributos e inclusive os comentários.

Todos eles são agrupados pelo DOM em um objeto Node. Elementos e textos são tipos especiais de nós mas também são nós. Quando tivermos um nó podemos capturar seu nome com nodeName e seu valor com nodeValue.

PERCORRENDO ÁRVORES DOM

Imagine que seu site é um documento HTML. O navegador (como o Chrome ou o Firefox) pega esse HTML e cria uma árvore de objetos — cada tag vira um nó (ou "node").

Assim, usando JavaScript, você pode:

- Acessar elementos (como parágrafos, imagens, botões);
- Alterar textos, cores, estilos;
- Criar novos elementos;
- Remover ou mover coisas na página;
- Reagir a ações do usuário (como cliques e digitação).

COMO É A ÁRVORE DOM?

Exemplo simples:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Minha Página</title>
  </head>
  <body>
    <h1>Bem-vindo!</h1>
    <p>Este é um parágrafo.</p>
  </body>
</html>
```

O navegador monta uma árvore assim:

- html
  - head
    - title
  - body
    - h1
    - P

Cada elemento é um **nó** (node) dentro da árvore.

EXEMPLOS PRÁTICOS

Mudar o texto de um título

```
<h1 id="meuParagrafo">Texto original</h1>
 <button onclick="mudarTexto()">Mudar Texto</button>

 <script>
  function mudarTexto() {
    document.getElementById("meuParagrafo").innerText = "Olá,
mundo!";
  }
 </script>
```

EXEMPLOS PRÁTICOS

Criar elementos dinamicamente

```
<input id="novoItem" type="text" placeholder="Novo item" />
    <button onclick="adicionarItem()">Adicionar</button>
    <ul id="lista"></ul>

    <script>
      function adicionarItem() {
        const texto = document.getElementById("novoItem").value;
        const li = document.createElement("li");
        li.innerText = texto;
        document.getElementById("lista").appendChild(li);
      }
    </script>
```

EXEMPLOS PRÁTICOS

Alterar a cor do fundo

```
<button onclick="fundoAzul()">Fundo Azul</button>
  <button onclick="fundoBranco()">Fundo Branco</button>

  <script>
    function fundoAzul() {
      document.body.style.backgroundColor = "blue";
    }

    function fundoBranco() {
      document.body.style.backgroundColor = "white";
    }
  </script>
```

EXEMPLOS PRÁTICOS

Contador de Cliques

```
<p id="contador">Você clicou 0 vezes.</p>
<button onclick="contar()">Clique para contar</button>

<script>
let cliques = 0;

function contar() {
  cliques++;
  document.getElementById('contador').innerText = 'Você clicou ' + cliques + '
vezes.';
}
</script>
```

ATÉ A PRÓXIMO!