DOM

**DOM** (Document Object Model), ou Modelo de Objeto de Documento, é uma API (Interface de Programação de Aplicações) que permite acessar e manipular a estrutura de um documento HTML ou XML de forma dinâmica.

O **DOM** transforma o código HTML em uma **árvore de objetos**, onde cada **elemento HTML** (como <div>, <p>, <h1>, etc.) é representado por um **nó** dessa árvore. Na estrutura hierárquica do DOM:

* O **elemento raiz** é o nó superior da árvore, que, em um documento HTML, é o elemento <html>.
* **Elementos pais** são nós que contêm um ou mais nós filhos. Por exemplo, o elemento <body> é pai de elementos como <h1> ou <p>.
* **Elementos filhos** são nós contidos dentro de outro nó. Por exemplo, o elemento <h1> é filho de <body>.

Essa estrutura em árvore facilita a navegação e manipulação do conteúdo da página, permitindo que o **JavaScript** interaja diretamente com a página da web. Com o **DOM**, é possível:

* **Acessar** os elementos da árvore, como buscar um elemento específico através de seus IDs, classes ou outras propriedades.
* **Modificar** o conteúdo ou atributos dos elementos, como alterar o texto de um parágrafo ou o valor de um atributo href.
* **Adicionar** novos elementos ou **remover** elementos existentes.

Em resumo, o **DOM** serve como a ponte entre o conteúdo HTML e o JavaScript, transformando o HTML em uma estrutura que pode ser manipulada de forma flexível e interativa. Isso permite que o desenvolvedor altere a página em tempo real sem precisar recarregá-la, criando experiências web dinâmicas e interativas.



**1. getElementById()**

* Esse método busca **um único elemento** a partir do **ID** especificado. O ID é um identificador único dentro do documento, o que significa que esse método sempre retornará apenas um elemento (ou null se não encontrar nenhum).
* O uso mais comum é para acessar elementos específicos, como um título ou um parágrafo, onde o ID foi previamente atribuído ao elemento.

**2. getElementsByClassName()**

* Esse método busca **todos os elementos** que possuem uma **classe específica**. Diferente do getElementById(), que retorna apenas um único elemento, este método pode retornar múltiplos elementos, já que várias tags HTML podem compartilhar a mesma classe.
* Ele é útil quando se deseja acessar todos os elementos de uma página que compartilham um estilo ou comportamento comum.

**3. getElementsByTagName()**

* Esse método busca **todos os elementos** de uma **tag específica**. Por exemplo, se você quiser acessar todos os parágrafos (<p>) ou todos os botões (<button>) em uma página, pode usar esse método.
* Assim como o getElementsByClassName(), ele retorna uma lista de elementos (ou uma coleção de nós), e não um único elemento.

**4. getElementsByName()**

* Esse método busca **todos os elementos** que possuem um atributo **name** específico. Esse atributo é frequentemente usado em elementos de formulários, como campos de entrada (inputs) e botões.
* É útil para selecionar todos os campos de um formulário com o mesmo nome, por exemplo, quando se tem vários campos de entrada com o mesmo nome de "usuário" ou "senha".