Por: Alice Dantas

DOM (Document Object Model)

É uma representação em forma de árvore de uma página web (HTML ou XML).

Quando você abre uma página no navegador: o navegador lê o HTML, transforma esse HTML em uma árvore de objetos e cada parte da página (parágrafos, títulos, imagens, botões) vira um nó (node) dentro dessa árvore. O JavaScript consegue acessar e modificar essa árvore. Isso permite que você **mude a página em tempo real**, sem precisar recarregar.

Estrutura de Árvore do DOM

Ex. de um HTML simples:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Exemplo DOM</title>
  </head>
  <body>
    <h1 id="titulo">Olá, mundo!</h1>
    <p class="texto">Esse é um parágrafo.</p>
    <button>Clique aqui</button>
  </body>
</html>
```

A árvore do DOM desse HTML fica assim:

```
Document
 └── html
      ├── head
      │    └── title ("Exemplo DOM")
      └── body
           ├── h1#titulo ("Olá, mundo!")
           ├── p.texto ("Esse é um parágrafo.")
           └── button ("Clique aqui")
```

Por: Alice Dantas

Como o JavaScript interage com o DOM

- Selecionando elementos:

```
document.getElementById("titulo");     // pelo id, modo antigo
document.querySelector("button");      // seleciona pela tag
document.querySelector(".texto");      // seleciona pela classe
document.querySelector("#titulo");     // seleciona pelo id
```

O `document.querySelector()` é uma forma **moderna** de buscar elementos no DOM. Ele **sempre pega só o primeiro** elemento que corresponder ao seletor. Para selecionar **todos os elementos**, use `document.querySelectorAll()`, que retorna uma **lista** de nós.

- Alterando conteúdo:

```
document.querySelector("#titulo").textContent = "<b>Novo</b> título!";
document.querySelector("#titulo").innerText = <b>Novo</b> título!";
```

Ps.: na página, o texto "Novo título!" é mostrado **literalmente**, ou seja, em ambos os casos, o navegador **não aplica negrito** — ele mostra as tags como texto comum. A diferença é que:

`textContent` → muda o **texto "real" armazenado na estrutura do navegador (DOM)**. Mesmo que o elemento esteja oculto, o texto muda.

`innerText` → muda o **texto que o usuário enxerga na tela**, respeitando regras de CSS como `display: none`, `visibility: hidden` e quebras de linha.

- Alterando estilo:

```
document.querySelector(".texto").style.color = "blue";
document.querySelector(“.texto”).style.fontSize = “32px”
```

- Adicionando e removendo classes:

```
const texto = document.querySelector(“.texto”)
texto.classList.add("ativo");
texto.classList.remove("inativo");
texto.classList.toggle("destaque"); // adiciona se não existir, remove
se existir
```

- Criando e Inserindo Elementos:

```
const novoParagrafo = document.createElement("p"); // cria um objeto do tipo <p>, mas ainda não aparece na página, existe só na memória.
novoParagrafo.textContent = "Seja bem-vindo"; // colocando algo dentro

document.body.appendChild(novoParagrafo); //insere ao fim da página
document.body.prepend(novoParagrafo); //insere no começo do pai

const titulo = document.querySelector("#titulo");
document.body.insertBefore(novoParagrafo, titulo); //insere antes de um outro elemento existente.
```

Você pode também adicionar **atributos** ao novo elemento:

```
const imagem = document.createElement("img");
imagem.src = "https://example.com/yoda.jpg";
imagem.alt = "Mestre Yoda";
document.body.appendChild(imagem);
```

Você pode criar **hierarquias** de elementos antes de inseri-los:

```
const div = document.createElement("div");
const titulo = document.createElement("h2");
const texto = document.createElement("p");

titulo.textContent = "Título dinâmico";
texto.textContent = "Texto gerado via JavaScript.";

div.appendChild(titulo);
div.appendChild(texto);

document.body.appendChild(div);
```

Isso gera:

```
<div>
  <h2>Título dinâmico</h2>
  <p>Texto gerado via JavaScript.</p>
</div>
```

- Removendo Elementos:

```
const paragrafo = document.querySelector("p");
paragrafo.remove();
```

- Reagindo a eventos (cliques, teclas, etc.):

```
document.querySelector("button").addEventListener("click", function()
{
  alert("Você clicou no botão!");
});
```

- Acessando Inputs e Formulários:

html

```
<input id="nome" type="text" placeholder="Digite seu nome">
<button id="enviar">Enviar</button>
<p id="mensagem"></p>
```

js

```
document.querySelector("#enviar").addEventListener("click", () => {
  const nome = document.querySelector("#nome").value;
  document.querySelector("#mensagem").textContent = `Olá, ${nome}!`;
});
```

O que acontece aqui:

Cada vez que o usuário clica no botão, o navegador executa novamente a função dentro do addEventListener. Dentro dessa função, você cria uma nova constante nome. Ela só existe dentro dessa execução (ou seja, dentro dessa chamada da função). Quando a função termina, essa const nome deixa de existir. Então, const nome não muda — ela é recriada do zero a cada clique.