

INFORMAÇÕES

MÓDULO	AULA	INSTRUTOR
04 JavaScript Essencial e Consumo de APIs	04 - DOM e Eventos	Pedro Henrique Amadio

CONTEÚDO

1. O que é o DOM?

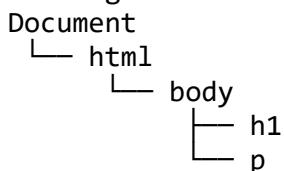
O DOM é uma **representação em forma de árvore** de todos os elementos de uma página web.

Quando o navegador carrega um arquivo HTML, ele **lê o código** e cria um modelo interno que organiza tudo como se fosse uma **árvore de objetos** - cada tag (como `<div>`, `<p>`, ``, etc.) vira um **nó (node)** nessa árvore.

Por exemplo, imagine esse HTML:

```
<html>
  <body>
    <h1>Título</h1>
    <p>Parágrafo de exemplo</p>
  </body>
</html>
```

O navegador converte isso em uma estrutura de árvore assim:



Cada um desses elementos (o `html`, o `body`, o `h1`, o `p`) é um **nó do DOM**.

Em JavaScript, o objeto global `document` representa toda essa estrutura.

Exemplo prático:

```
console.log(document.title); // Título da página
console.log(document.body); // Mostra o corpo do HTML
console.log(document.URL); // Exibe o endereço atual
```

Dica:

Sempre que o navegador carrega uma página, ele cria uma cópia “viva” da estrutura HTML no DOM.

Quando você altera o DOM com JavaScript, o conteúdo da página muda imediatamente – sem precisar recarregar.

2. Estrutura do DOM

A árvore do DOM é composta por **nós (nodes)**.

Existem vários tipos de nós, mas os principais são:

TIPO DE NÓ	DESCRIÇÃO	EXEMPLO
Document	Representa o documento inteiro. É o ponto de entrada para o DOM.	document
Element	Representa uma tag HTML.	<div>, <p>, <button>
Text	Representa o texto dentro de um elemento.	“Olá, mundo!”
Attribute	Representa um atributo de uma tag.	id=“titulo”, class=“destaque”

3. Seletores e Acesso a Elementos

Para manipular o DOM, o primeiro passo é selecionar elementos. O JavaScript oferece várias formas de encontrar um elemento.

MÉTODO	O QUE FAZ	RETORNA
<code>document.getElementById(“id”)</code>	Busca pelo atributo id	Um único elemento
<code>document.getElementsByClassName(“classe”)</code>	Busca por classe	Lista (HTMLCollection)
<code>document.getElementsByTagName(“div”)</code>	Busca por tag	Lista (HTMLCollection)
<code>document.querySelector(“seletor”)</code>	Seleciona o primeiro que corresponder ao seletor CSS	Um elemento
<code>document.querySelectorAll(“seletor”)</code>	Seleciona todos os que corresponderem	Lista (NodeList)

Exemplo:

```

<p id=“mensagem”>Olá!</p>
<p class=“texto”>Bem-vindo!</p>

<script>
  const msg = document.getElementById(“mensagem”);
  const textos = document.getElementsByClassName(“texto”);
  const primeiroParagrafo = document.querySelector(“p”);
  const todosParagrafos = document.querySelectorAll(“p”);

  console.log(msg.textContent); // “Olá!”
  console.log(textos[0].textContent); // “Bem-vindo!”
</script>

```

3.1. Métodos Clássicos de Seleção

- `getElementById()`
Busca um único elemento pelo seu atributo id.

```
const elemento = document.getElementById("mensagem");
console.log(elemento.textContent); // "Olá, visitante!"
```

Observação:

IDs devem ser únicos dentro da página (discutimos sobre, no nosso módulo) - portanto, esse método sempre retorna **apenas um** elemento.

- **getElementsByClassName()**

Busca **todos os elementos** que tenham a mesma classe.

```
<p class="texto">Um</p>
<p class="texto">Dois</p>
```

```
const elementos = document.getElementsByClassName("texto");
console.log(elementos[0].textContent); // "Um"
```

Retorna uma **coleção viva (HTMLCollection)** – ou seja, se novos elementos com essa classe forem adicionados depois, eles **aparecem automaticamente** nessa coleção.

- **getElementsByTagName()**

Busca **todos os elementos** com uma determinada tag, como div, p, img, etc.

```
const paragrafos = document.getElementsByTagName("p");
console.log(paragrafos.length); // quantidade de <p> na página
```

Também retorna uma **HTMLCollection**.

Modernos

3.2. Métodos Clássicos de Seleção

Os métodos modernos do DOM usam a **mesma sintaxe dos seletores CSS**, o que os torna mais poderosos e fáceis de entender.

1. querySelector()

Seleciona o **primeiro elemento** que corresponde ao seletor CSS especificado.

```
<div class="caixa destaque"></div>

const caixa = document.querySelector(".caixa.destaque");
```

Você pode usar:

- **Classes:** .nomeDaClasse
- **IDs:** #nomeDoId
- **Tags:** div, p, button
- **Seletores combinados:** div p, ul li, header nav a

2. querySelectorAll()

Seleciona **todos os elementos** que correspondem ao seletor CSS informado.

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
</ul>

const itens = document.querySelectorAll("ul li");
itens.forEach((li) => console.log(li.textContent));
```

Retorna um **NodeList**, que permite usar métodos como forEach().

3.3. Hierarquia e Seleção de Filhos/Pais

O DOM é uma árvore, e muitas vezes você precisa navegar entre pais, filhos e irmãos.

3. Selecionar filho

```
const lista = document.querySelector("ul");
const primeiroItem = lista.firstChild; // primeiro <li>
```

4. Selecionar pai

```
const item = document.querySelector("li");
console.log(item.parentElement); // retorna o <ul> pai
```

5. Selecionar irmão

```
const item = document.querySelector("li");
console.log(item.nextElementSibling); // próximo <li>
```

3.4. Seletores Compostos

Você pode usar vários critérios juntos, como no CSS:

EXEMPLO	SELEICIONA
"div p"	Todos os <p> dentro de <div>
"ul > li"	Somente os filhos diretos de
".menu a"	Todos os <a> dentro de elementos com classe menu
"[data-id]"	Todos os elementos com o atributo data-id
"[type='text']"	Todos os <input> cujo tipo seja texto

Exemplo:

```
const botoes = document.querySelectorAll("button[data-action='enviar']);
```

4. Manipulação de Conteúdo e Estilos

Após selecionar um elemento, podemos alterar seu conteúdo, atributos e estilo.

6. Alterando conteúdo:

```
const titulo = document.querySelector("h1");
titulo.textContent altera apenas o texto bruto;
titulo.innerHTML = "<em>Título em itálico!</em>";
innerHTML permite inserir HTML dentro do elemento.
```

7. Alterando estilos (CSS inline)

```
titulo.style.color = "blue";
titulo.style.backgroundColor = "lightgray";
titulo.style.fontSize = "32px";titulo.textContent = "Título alterado!"; // altera o texto visível
```

O style altera apenas o estilo inline, diretamente no elemento.

8. Alterando atributos

```
const link = document.querySelector("a");
link.setAttribute("href", "https://developer.mozilla.org");
link.setAttribute("target", "_blank");
```

Para remover:

```
link.removeAttribute("target");
```

9. Manipulando classes

```
const box = document.querySelector(".box");

box.classList.add("ativo"); // adiciona uma classe
box.classList.remove("inativo"); // remove uma classe
box.classList.toggle("destaque"); // alterna entre adicionar e remover
```

Uso prático:

```
<div class="caixa">Texto original</div>
<button onclick="alterar()">Alterar</button>

<script>
  function alterar() {
    const caixa = document.querySelector(".caixa");
    caixa.textContent = "Texto alterado!";
    caixa.classList.toggle("ativo");
  }
</script>
```

5. Criação e Remoção de Elementos

O DOM permite criar novos elementos dinamicamente.

- **Criar elementos:**

```
const novoItem = document.createElement("li");
novoItem.textContent = "Novo item!";
document.querySelector("ul").appendChild(novoItem);
```

- **Inserir antes de outro elemento:**

```
const lista = document.querySelector("ul");
const item = document.createElement("li");
item.textContent = "Primeiro item!";
lista.prepend(item); // insere no início
```

- **Remover elementos:**

```
const item = document.querySelector("li");
item.remove(); // remove o primeiro <li>
```

Essas técnicas serão essenciais quando começarmos a renderizar dados vindos de APIs (na Aula 06).

6. Introdução a Eventos

Eventos são ações que acontecem na página e que podemos “escutar” com JavaScript.

Exemplos de eventos:

- click → Clique do mouse
- mouseover → Passar o mouse sobre algo
- keydown → Pressionar uma tecla

- `input` → Quando o usuário digita
- `submit` → Envio de formulários

- 3 formas de associar eventos:

- `Inline (HTML):`

```
<button onclick="alert('Clicado!')>Clique aqui</button>
```

⚠ Evite usar inline. Mistura HTML e JS e dificulta manutenção.

- `No script (propriedade direta):`

```
const botao = document.querySelector("button");
botao.onclick = function() {
    alert("Clicado!");
};
```

⚠ Sobrescreve o evento anterior se for atribuído de novo.

- `addEventListener (moderno e recomendado):`

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

✓ Permite adicionar vários ouvintes para o mesmo evento.

✓ Mantém o HTML limpo e o código mais modular.

7. Eventos e Manipulação Interativa

Exemplo prático:

```
<h2 id="titulo">Bem-vindo!</h2>
<button id="btn">Clique aqui</button>

<script>
    const titulo = document.getElementById("titulo");
    const botao = document.getElementById("btn");

    botao.addEventListener("click", () => {
        titulo.textContent = "Você clicou!";
        titulo.style.color = "green";
    });
</script>
```

- Eventos de teclado:

```
<input id="campo" placeholder="Digite algo..." />

<script>
    const campo = document.getElementById("campo");
    campo.addEventListener("input", (e) => {
        console.log("Valor atual:", e.target.value);
    });
</script>
```

`e.target.value` pega o valor do campo no momento da digitação.

8. Projeto Prático - Lista de Compras + Contador Dinâmico

- Parte 1 - Lista de compras

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <title>Lista de Compras</title>
</head>
<body>
    <h2>🛒 Minha Lista de Compras</h2>

    <input id="item" placeholder="Digite um produto..." />
    <button id="adicionar">Adicionar</button>

    <ul id="lista"></ul>

    <script>
        const input = document.getElementById("item");
        const botao = document.getElementById("adicionar");
        const lista = document.getElementById("lista");

        botao.addEventListener("click", () => {
            const produto = input.value.trim();

            if (produto === "") {
                alert("Digite o nome de um produto!");
                return;
            }

            const li = document.createElement("li");
            li.textContent = produto;
            lista.appendChild(li);

            input.value = "";
            input.focus();
        });
    </script>
</body>
</html>
```

- Parte 2 - Contador Interativo

```

<h2 id="contador">0</h2>
<button id="mais">+</button>
<button id="menos">-</button>

<script>
  let valor = 0;
  const contador = document.getElementById("contador");

  document.getElementById("mais").addEventListener("click", () => {
    valor++;
    contador.textContent = valor;
  });

  document.getElementById("menos").addEventListener("click", () => {
    valor--;
    contador.textContent = valor;
  });
</script>

```

9. Exercícios de Fixação

1. O que é o DOM em JavaScript?

- a) Uma biblioteca de animações.
- b) A estrutura em árvore de um documento HTML.
- c) Um banco de dados interno do navegador.
- d) Um framework JavaScript.

2. Qual método seleciona um elemento pelo ID?

- a) document.querySelector(".id")
- b) document.getElement("id")
- c) document.getElementById("id")
- d) document.selectId("id")

3. O que o método addEventListener() faz?

- a) Remove um evento do elemento.
- b) Adiciona um evento sem precisar alterar o HTML.
- c) Cria um novo elemento no DOM.
- d) Muda o valor de um atributo.

4. O que o código abaixo faz?

```

const p = document.querySelector("p");
p.textContent = "Olá Mundo!";

```

- a) Cria um novo parágrafo.
- b) Altera o texto do primeiro <p>.
- c) Remove o parágrafo.
- d) Duplica o elemento.

5. O método .classList.toggle("ativo"):

- a) Adiciona uma classe apenas se ela não existir.
- b) Remove todas as classes.
- c) Alterna a presença da classe "ativo".
- d) Só funciona em eventos de teclado.

10. Para Casa

Crie uma página simples de **lista de tarefas (To-Do List)** com os seguintes requisitos:

1. Um campo de texto (input) e um botão “Adicionar”.
2. Cada nova tarefa digitada deve ser adicionada a uma lista (ul).
3. Cada item deve ter um botão “Remover”.
4. Um botão “Limpar tudo” deve esvaziar a lista inteira.

Dica: use os métodos:

- createElement()
- appendChild()
- remove() ou innerHTML = “”

Desafio extra (opcional):

Adicione um contador que mostra quantas tarefas existem na lista.