



# Elementos do DOM


**H**oje vamos explorar um dos conceitos fundamentais para o desenvolvimento web dinâmico: os Elementos do DOM (Document Object Model). O DOM é a ponte que conecta o conteúdo estático de uma página HTML com a interatividade que o JavaScript pode proporcionar. Ao entender como o DOM organiza os elementos da página em uma estrutura hierárquica, seremos capazes de selecionar, modificar, criar e manipular esses elementos de maneira eficiente.

## Definição de DOM

O DOM, ou Modelo de Objeto de Documento, é uma interface de programação que permite ao JavaScript interagir com documentos HTML e XML, embora, no nosso contexto, vamos trabalhar especificamente com HTML. O DOM pode ser visualizado como uma estrutura hierárquica, onde cada elemento do HTML, como parágrafos, imagens, tabelas, e até mesmo atributos e textos, são representados como objetos em uma árvore. Essa estrutura em árvore permite que cada parte do documento seja acessada e manipulada de forma programática.

Por exemplo, ao visualizar uma página HTML, o DOM permite que você selecione um parágrafo específico e altere seu conteúdo ou estilo. Isso é possível porque o parágrafo é representado como um nó na árvore do DOM, e o JavaScript pode acessar esse nó e modificar suas propriedades. Outro exemplo prático seria a criação de novos elementos na página, como adicionar um botão ou uma imagem, tudo isso através da manipulação do DOM.

A importância de aprender sobre o DOM reside no fato de que ele é fundamental para a manipulação dinâmica de páginas web. Sem o entendimento do DOM, seria muito mais difícil interagir com os elementos de uma página de maneira eficaz e

eficiente. Além disso, compreender essa estrutura facilita o aprendizado de outras tecnologias e frameworks que também utilizam o DOM como base para manipulação de documentos. 


Ao longo dos próximos tópicos, vamos nos aprofundar nas práticas de manipulação de elementos no DOM, explorando como localizar esses elementos dentro da árvore, criar novos elementos, e realizar diversas outras interações que tornarão nossas páginas mais dinâmicas e interativas. Vamos também utilizar o navegador para experimentar e ver na prática como essas manipulações acontecem em tempo real, consolidando assim o aprendizado teórico com a prática.

## Usando DOM

Nesta segunda parte da nossa aula, vamos aprofundar o uso do DOM e explorar como utilizá-lo de forma eficiente para manipular os elementos de uma página web. Um dos aspectos mais importantes ao trabalhar com o DOM é a capacidade de identificar e acessar cada elemento que desejamos manipular. Existem várias maneiras de fazer isso, e cada uma traz benefícios específicos dependendo do que precisamos alcançar.

Para acessar um elemento do DOM, por exemplo, podemos utilizar métodos que nos permitem selecionar um elemento específico na árvore de objetos. Uma vez identificado, é possível manipular o conteúdo desse elemento, alterando, por exemplo, o texto exibido ou o valor de um campo. Além disso, podemos modificar atributos do elemento, como adicionar novos atributos ou alterar os existentes, e também manipular seus estilos visuais, como a cor do texto ou do fundo, usando CSS diretamente através do DOM.

Outra funcionalidade poderosa do DOM é a capacidade de adicionar ou remover classes de um elemento, o que nos permite alterar a aparência de um elemento de maneira dinâmica. Podemos também criar e remover elementos da árvore do DOM, o que é especialmente útil quando precisamos adicionar novos componentes na página ou remover elementos que não são mais necessários.

Além das manipulações básicas, o DOM também nos permite trabalhar com eventos, como cliques de mouse ou interações de teclado. Por exemplo, poder  definir ações específicas para eventos como um clique duplo, pressionar uma tecla ou desfocar um campo de texto. Isso nos dá um controle preciso sobre como os usuários interagem com a página e como essas interações afetam os elementos da interface.

Um exemplo prático de uso do DOM é quando estamos desenvolvendo uma página e precisamos visualizar como uma alteração no código HTML se refletirá na interface do usuário. Podemos inspecionar e modificar elementos diretamente no navegador, testar diferentes estilos e ver o resultado em tempo real. Por exemplo, podemos alterar a cor de um botão ou o texto de uma célula de tabela, tudo isso diretamente na visualização do DOM no navegador. Se o resultado for satisfatório, podemos então aplicar essas alterações permanentemente no código HTML.

Esse tipo de manipulação em tempo real facilita muito o processo de desenvolvimento, permitindo que experimentemos diferentes soluções antes de implementá-las definitivamente no código. É uma forma prática e eficaz de garantir que o comportamento visual e funcional da página esteja alinhado com o esperado.

## **Encontrando elementos**

Nesta terceira parte da nossa aula, vamos explorar como encontrar elementos no DOM utilizando código JavaScript. O primeiro passo para manipular elementos é aprender a localizá-los de maneira eficiente. O JavaScript nos oferece diversas funções para facilitar essa tarefa, sendo as principais: o `'getElementById'`, `'getElementsByClassName'`, `'getElementsByTagName'`, e o `'querySelector'`. Cada uma dessas funções tem seu uso específico, e vamos focar nas duas mais comuns: `getElementById` e `getElementsByClassName`.

A função `'getElementById'` é a mais utilizada e importante. Ela nos permite encontrar um elemento específico no DOM com base em seu identificador único (ID). Lembre-se de que o ID deve ser único para cada elemento, o que garante que sempre selecionaremos o elemento correto ao usar essa função. Com o elemento

encontrado, podemos realizar diversas operações, como alterar seu conteúdo, modificar atributos ou até mesmo removê-lo.



Outra função útil é a `'getElementsByClassName'`, que nos permite selecionar todos os elementos que compartilham uma mesma classe. Diferente do ID, uma classe pode ser atribuída a vários elementos, permitindo que façamos alterações em grupo. Por exemplo, podemos selecionar todas as tabelas que têm uma determinada classe e aplicar estilos ou outras modificações a elas.


Para ilustrar, imagine que você tenha duas tabelas em uma página HTML, uma para produtos e outra para formulários, e ambas possuem classes específicas. Podemos usar o `'getElementsByClassName'` para encontrar e manipular essas tabelas. No console do navegador, ao digitar `'document.getElementsByClassName('tabela-produto')'`, o JavaScript retornará todos os elementos com essa classe, permitindo que você altere, por exemplo, o estilo ou o conteúdo das células da tabela de produtos.

Além disso, ao usar o `'getElementById'`, podemos selecionar elementos individuais como campos de formulário específicos e modificá-los conforme necessário. Se quisermos alterar o texto de um campo identificado por `"descricao-produto"`, basta usar `'document.getElementById("descricao-produto")'` para localizá-lo e, em seguida, aplicar as modificações desejadas.

Essas operações mostram a importância de atribuir IDs e classes de forma estratégica ao desenvolver uma página web. Usar nomes sugestivos facilita a localização e manipulação dos elementos no DOM, tornando o desenvolvimento mais eficiente e organizado.

## **Criando Elementos**

Agora, vamos explorar como criar novos elementos no DOM utilizando JavaScript. A capacidade de criar, remover e manipular elementos dinamicamente é uma das principais vantagens ao trabalhar com o DOM, pois permite que nossas páginas web sejam mais interativas e dinâmicas.

Para ilustrar o processo de criação de elementos, vamos usar um exemplo prático. Suponha que queremos adicionar uma nova linha em uma tabela HTML  existente. Para isso, o primeiro passo é criar o novo elemento utilizando o método 'createElement'. Nesse caso, queremos adicionar uma nova linha, então utilizamos 'document.createElement('tr')' para criar esse elemento de linha, que é atribuído a uma variável chamada 'novaLinha'.

Em seguida, precisamos criar as células que irão compor essa nova linha. Utilizamos o mesmo método 'createElement', mas desta vez para criar elementos 'td', que representam as células. Cada célula é atribuída a uma variável, por exemplo, 'novaCelula1' e 'novaCelula2'. Depois, usamos a propriedade innerText para definir o texto que cada célula irá exibir, como "TextoCélula1" e "TextoCélula2".


Após criar as células, precisamos inseri-las dentro da linha. Isso é feito com o método 'appendChild', que adiciona as células como "filhas" da linha. Repetimos o processo para cada célula, garantindo que ambas sejam inseridas na linha.

Agora que a linha está completa com suas células, o próximo passo é localizar a tabela onde queremos inserir essa linha. Utilizamos 'getElementById' para selecionar a tabela pelo seu ID e, em seguida, usamos novamente 'appendChild' para adicionar a nova linha na tabela.

Ao salvar e executar esse código no navegador, veremos que a nova linha é inserida na tabela, com as células contendo os textos definidos. Esse exemplo mostra como o DOM permite criar novos elementos de forma dinâmica e controlada, proporcionando uma grande flexibilidade no desenvolvimento de páginas web.

A criação de novos elementos no DOM é uma prática essencial para tornar as páginas mais interativas e responsivas com as ações do usuário. Ao dominar essa técnica, conseguimos manipular a estrutura do documento HTML de forma eficaz, adicionando novos componentes conforme necessário e melhorando a experiência do usuário.

## Explorando o uso de Elementos


Nesta última parte da nossa aula sobre "Elementos do DOM", vamos explorar como utilizar e manipular elementos no DOM de forma mais avançada. Existem diversas operações que podemos realizar, como alterar o texto de um elemento, modificar o HTML diretamente e manipular atributos desses elementos. 

Para alterar o texto de um elemento, podemos usar as propriedades `textContent` e `innerText`. Ambas permitem que você modifique o conteúdo textual de um elemento, mas é importante notar que `textContent` considera todo o texto, incluindo o que está oculto, enquanto `innerText` leva em conta apenas o texto visível. Já para alterar o HTML interno de um elemento, utilizamos a propriedade `innerHTML`. Atribuindo um novo valor a `innerHTML`, você pode modificar todo o conteúdo HTML de um elemento, incluindo a estrutura interna.

Além de manipular o conteúdo, também é possível trabalhar com atributos dos elementos utilizando os métodos `getAttribute` e `setAttribute`. O método `getAttribute` permite que você obtenha o valor de um atributo específico de um elemento, enquanto `setAttribute` permite que você altere ou crie um novo atributo para esse elemento. Essas operações são fundamentais para controlar o comportamento e a aparência dos elementos dinamicamente.

Vamos ver um exemplo prático de como isso funciona. Suponha que você tenha um campo de entrada de texto em uma página, identificado pelo ID `"codigoProduto"`. Usando `document.getElementById('codigoProduto')`, você pode acessar esse elemento e atribuir sua referência a uma variável para facilitar a manipulação. Se quiser alterar o tipo desse campo para aceitar apenas números, você pode usar `setAttribute('type', 'number')`, o que fará com que o campo aceite apenas entradas numéricas. Além disso, você pode adicionar um novo atributo, como `'step'`, que define os incrementos de valor permitidos para números, e configurá-lo para 0.01, permitindo incrementos decimais.

Essas operações podem ser realizadas tanto diretamente no console do navegador para testes rápidos quanto no código JavaScript para que as alterações sejam permanentes na aplicação. Por exemplo, se você quiser garantir que um campo de

formulário aceite apenas números e tenha incrementos específicos, pode aplicar essas alterações no código JavaScript que será executado sempre que a página  carregar.

Compreender como explorar o uso dos elementos no DOM é essencial para criar páginas interativas e dinâmicas. Manipular texto, HTML e atributos abre um leque de possibilidades para customizar e controlar o comportamento dos elementos conforme as necessidades do projeto.

## **Conteúdo Bônus**

Para auxiliar nos estudos sobre operadores, recomendo assistir ao vídeo “Curso JavaScript #35 - Acessando elementos através do DOM”, que está disponível no canal Matheus Battisti - Hora de Codar no YouTube.

## **Referência Bibliográfica**

ASCENCIO, A. F. G.; CAMPOS, E. A. V. de. Fundamentos da programação de computadores. 3.ed. Pearson: 2012.

BRAGA, P. H. Teste de software. Pearson: 2016.

GALLOTTI, G. M. A. (Org.). Arquitetura de software. Pearson: 2017.

GALLOTTI, G. M. A. Qualidade de software. Pearson: 2015.

MEDEIROS, E. Desenvolvendo software com UML 2.0 definitivo. Pearson: 2004.

PFLEEGER, S. L. Engenharia de software: teoria e prática. 2.ed. Pearson: 2003.

SOMMERVILLE, I. Engenharia de software. 10.ed. Pearson: 2019.



### Atividade Prática 12 – Interação e Manipulação Dinâmica com o DOM

#### Objetivos:

- Compreender e aplicar conceitos de manipulação de elementos do DOM.
- Utilizar métodos do DOM para selecionar, criar e modificar elementos de uma página web.
- Explorar técnicas avançadas para alterar o conteúdo, atributos e estrutura dos elementos no DOM.

#### Materiais, Métodos e Ferramentas:

- Computador com um editor de código (Visual Studio Code, Sublime Text, etc.)
- Navegador web (para executar e testar o código JavaScript)
- Acesso a uma página HTML básica para testes (ou crie uma nova página HTML)

#### Atividade Prática

Primeiramente, leia atentamente o texto a seguir:

Nesta atividade, você irá explorar a manipulação de elementos do DOM, incluindo a criação e modificação de elementos, bem como a manipulação de seus atributos e conteúdo. A atividade envolve a prática de seleção de elementos, criação de novos elementos e alterações em tempo real no DOM.

Agora, vamos praticar!



## PASSO A PASSO DETALHADO DA ATIVIDADE:



- **Manipulação Básica de Elementos:**


- Crie um arquivo HTML e inclua um arquivo JavaScript.
- No arquivo HTML, adicione um parágrafo com o ID “mensagem” e um botão com o ID “mudarTexto”.
- No arquivo JavaScript, adicione um código para selecionar o parágrafo usando getElementById e altere seu conteúdo para “Texto Original”.
- Adicione um evento ao botão que, quando clicado, altera o texto do parágrafo para “Texto Alterado”.

- **Criando e Adicionando Novos Elementos:**

- No arquivo JavaScript, crie um novo elemento de lista (li) com o texto “Novo Item”.
- Adicione esse novo item a uma lista existente no HTML com o ID “lista”.
- Certifique-se de que o novo item seja exibido corretamente na página.


- **Modificando Atributos e Conteúdo:**

- Adicione um campo de entrada de texto com o ID “codigoProduto” no arquivo HTML.
- No arquivo JavaScript, altere o atributo type do campo para “number” usando setAttribute e adicione o atributo step com o valor “0.01”.

- Altere o texto do campo de entrada para “Insira um número” utilizando placeholder. 
- **Explorando Métodos Avançados do DOM:**
  - No arquivo JavaScript, selecione todos os elementos com a classe “item” utilizando `getElementsByClassName`.
  - Modifique o estilo de todos os elementos selecionados, alterando a cor do texto para “red”.
  - Adicione um novo elemento com a classe “item” à página e verifique se o estilo é aplicado corretamente.

## Gabarito Esperado

- **Manipulação Básica de Elementos:** o parágrafo com ID “mensagem” deve ser corretamente selecionado e alterado pelo botão, com o texto atualizado conforme esperado.
- **Criando e Adicionando Novos Elementos:** o novo item deve ser adicionado à lista existente com sucesso e visível na página.
- **Modificando Atributos e Conteúdo:** o campo de entrada deve ter o atributo `type` alterado para “number” e o atributo `step` adicionado, com o texto de placeholder atualizado.
- **Explorando Métodos Avançados do DOM:** todos os elementos com a classe “item” devem ter o estilo alterado e um novo elemento com a mesma classe deve ser adicionado, com o estilo aplicado corretamente.

Certifique-se de que todas as operações e manipulações sejam realizadas conforme especificado para garantir uma prática eficaz e a compreensão completa dos conceitos abordados. 

**Ir para exercício**