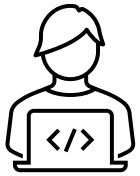
Prof. Lucas Teixeira

Repositório de aulas:
bit.ly/4bHLaad



Front-End

Desvendando o DOM: O que é e por que é importante?

O DOM, sigla para **Document Object Model**, representa a estrutura de um documento HTML ou XML como uma árvore hierárquica de objetos. Essa representação permite que linguagens de script, como o JavaScript, interajam e modifiquem dinamicamente o conteúdo da página, tornando-a mais viva e responsiva.

Desvendando o DOM: O que é e por que é importante?

Imagine o DOM como um modelo em miniatura da sua página web, onde cada elemento HTML se torna um objeto com propriedades e métodos específicos. Através do DOM, você pode:

- **Selecionar elementos**: Buscar elementos específicos na página, como parágrafos, imagens ou botões, usando seletor CSS ou outros métodos.

Desvendando o DOM: O que é e por que é importante?

- **Modificar conteúdo**: Alterar o texto, imagens ou outros dados dentro dos elementos selecionados, criando páginas dinâmicas e interativas.

- **Manipular estilos**: Aplicar estilos CSS aos elementos, controlando cores, fontes, tamanhos e outros aspectos visuais da página.

- **Responder a eventos**: Criar ações em resposta a eventos do usuário, como cliques, digitações ou movimentos do mouse, tornando a página interativa.

Desvendando o DOM: O que é e por que é importante?

Em resumo, o **DOM** é a chave para dar vida à sua página web, transformando-a em uma experiência rica e interativa para seus usuários.

Estrutura e Navegação

A estrutura do **DOM** é como uma árvore genealógica invertida, com o elemento raiz no topo e os elementos filhos ramificando-se abaixo dele. Cada elemento possui propriedades e métodos que permitem acessá-lo, modificá-lo e navegar pela árvore.

- **Elemento Raiz**: O documento HTML em si é o elemento raiz, representado pelo objeto document.

Estrutura e Navegação

- **Elementos Filhos**: Cada elemento pode conter outros elementos aninhados, criando uma hierarquia.
- **Propriedades**: Cada elemento possui propriedades que armazenam informações sobre si mesmo, como seu conteúdo, estilo e atributos.
- **Métodos**: Cada elemento possui métodos que permitem realizar ações sobre si mesmo, como adicionar ou remover conteúdo, alterar estilos ou responder a eventos.

Estrutura e Navegação

Para navegar pela árvore do DOM, podemos utilizar diversas técnicas:

- **getElementById**: Seleciona um elemento pelo seu ID único.
- **getElementsByTagName**: Seleciona todos os elementos com uma determinada tag HTML.
- **getElementsByClassName**: Seleciona todos os elementos com uma determinada classe CSS.

Estrutura e Navegação

- **querySelector**: Seleciona um elemento usando um seletor CSS mais complexo.
- **querySelectorAll**: Seleciona todos os elementos que correspondem a um seletor CSS mais complexo.

Ao dominar essas técnicas de navegação, você terá total controle sobre os elementos da sua página e poderá realizar manipulações complexas com maestria.

Estrutura e Navegação

```
// Selecionando o primeiro parágrafo da página
const paragraph = document.querySelector('p');

// Acessando o conteúdo do parágrafo
const paragraphContent = paragraph.textContent;

// Alterando o conteúdo do parágrafo
paragraph.textContent = 'Este parágrafo foi modificado pelo
JavaScript!';

// Selecionando todos os elementos com a classe "button"
const buttons = document.querySelectorAll('.button');

// Adicionando um evento de clique a cada botão
buttons.forEach(button => {
  button.addEventListener('click', () => {
    alert('Você clicou no botão!');
  });
});
```

Manipulando Conteúdo

O DOM torna possível modificar o conteúdo de um documento HTML de forma dinâmica, utilizando JavaScript. Isso abre um mundo de possibilidades para criar páginas interativas e personalizadas:

- **Alterar Texto**: Você pode alterar o texto dentro de qualquer elemento HTML, como parágrafos, títulos ou legendas.
- **Inserir Imagens**: Dinamicamente adicionar e remover imagens na página, criando galerias interativas ou carregando imagens sob demanda.
- **Modificar Estilos**: Aplicar estilos CSS aos elementos em tempo real, alterando cores, fontes, tamanhos e outros aspectos visuais da página.

Manipulando Conteúdo

Através dessas técnicas de manipulação, você poderá transformar a aparência e o conteúdo da sua página de acordo com as necessidades do usuário ou com base em dados externos.

```
const image = document.getElementById('myImage');

// Alterando a fonte da imagem
image.src = 'https://nova-imagem.jpg';

// Aplicando um estilo CSS à imagem
image.style.width = '200px';
image.style.height = 'auto';
```

Eventos

Um dos recursos mais poderosos do DOM é a capacidade de responder a eventos do usuário, como cliques, digitações ou movimentos do mouse. Isso permite que você crie páginas interativas que reagem às ações do usuário em tempo real:

Eventos

Eventos de Clique: Adicione ações a serem executadas quando um usuário clica em um elemento, como redirecioná-lo para outra página ou exibir um pop-up.

Eventos de Digitação: Capture a entrada do usuário em campos de texto e personalize a página de acordo com o que ele digita.

Eventos de Movimento do Mouse: Crie efeitos visuais dinâmicos ou navegue por menus interativos com base na movimentação do mouse do usuário

Eventos

```
const button = document.getElementById('myButton');

button.addEventListener('click', () => {
  alert('Você clicou no botão!');

  const element = document.getElementById('myElement');
  element.style.display = element.style.display === 'none' ?
  'block' : 'none';
});
```

Criando Elementos e Construindo Conteúdo Dinâmico

O DOM também permite criar novos elementos HTML e inseri-los na página de forma dinâmica, abrindo um leque de possibilidades para interfaces interativas e conteúdo personalizado:

- **Criar Elementos**: Utilize métodos como **document.createElement( )** para gerar novos elementos HTML, como parágrafos, imagens ou botões.

- Inserir Elementos: Posicione os elementos recém-criados em qualquer lugar da árvore do DOM, utilizando métodos como **appendChild( )** ou **insertBefore( ).**

- Remover Elementos: Elimine elementos da página quando necessário, utilizando o método **removeChild( ).**

Criando Elementos e Construindo Conteúdo Dinâmico

```
// Criando um novo parágrafo
const paragraph = document.createElement('p');

// Adicionando texto ao parágrafo
paragraph.textContent = 'Este parágrafo foi criado dinamicamente
pelo JavaScript!';

// Selecionando o elemento pai onde o parágrafo será inserido
const parentElement = document.getElementById('myContainer');

// Inserindo o parágrafo no elemento pai
parentElement.appendChild(paragraph);
```

Estilos CSS e Manipulação Dinâmica

O DOM permite aplicar estilos CSS aos elementos de forma dinâmica, controlando a aparência da página em tempo real e criando efeitos visuais interativos:

- **Propriedade style**: Cada elemento possui uma propriedade style que fornece acesso aos seus estilos CSS.

- **Definindo Estilos**: Você pode definir valores para propriedades CSS diretamente na propriedade style, alterando cores, fontes, tamanhos e outros aspectos visuais do elemento.

Estilos CSS e Manipulação Dinâmica

- **Classes CSS**: Utilize classes CSS para agrupar elementos com estilos semelhantes e aplique-as ou remova-as dinamicamente usando o método classList.

  Exemplo

```
const element = document.getElementById('myElement');

element.style.backgroundColor = 'red';

element.classList.add('active');

element.classList.remove('hidden');
```

Bibliografia

Online

Mozilla: https://developer.mozilla.org/pt-BR/docs/Web/HTML

W3C Schools:
https://www.w3schools.com/html/

Livros

Html 5: Entendendo e Executando:
https://a.co/d/9o3XsHS
CSS Cookbook: *https://a.co/d/6eU6USL*
JavaScript: O Guia Definitivo*: https://a.co/d/342B0rJ*
HTML5 e CSS3: Guia Prático e Visual:
https://a.co/d/jg3ccdP
Use a Cabeça! HTML e CSS: *https://a.co/d/66caZxW*
A psicologia das cores: *https://a.co/d/1jVHsnO*

Por Hoje é Só

Continuamos na próxima aula

Repositório de aulas:
https://bit.ly/4bHLaad