

O que é DOM em JavaScript ?

O DOM (Document Object Model) é uma interface de programação que representa a estrutura de um documento HTML, XML ou SVG como uma árvore de nós. Cada nó representa um elemento, atributo ou texto do documento.

O JavaScript pode interagir com o DOM para modificar a estrutura, o estilo e o conteúdo de um documento. Isso permite criar páginas web dinâmicas e interativas.

Por exemplo, você pode usar o JavaScript para:

- **Modificar o conteúdo de um elemento:** alterar o texto de um parágrafo, adicionar ou remover elementos de uma lista, etc.
- **Alterar o estilo de um elemento:** Modificar a cor, o tamanho da fonte, a posição, etc.
- **Adicionar eventos a um elemento:** Fazer com que um elemento reaja a ações do usuário, como cliques, movimentos do mouse, etc.

Aqui está um exemplo simples de como modificar o conteúdo de um elemento usando JavaScript:

O termo **DOM** em JavaScript significa **Modelo de Objeto de Documento**,

O DOM permite que você:

1. **Acesse** os `<div>`, ```<button>`, `<input>`, etc.).
2. **Modifique** o conteúdo ou
3. **Adicione** ou **remover** elementos HTML.
4. **Alterar** atributos de
5. **Escudo** eventos

Como funciona

O DOM representa

Selecionar elementos HTML :

eu

```
let elemento = document.getElementById('minhaDiv'); // Seleciona um elemento com id 'minhaDiv'
```

1.

Alterar conteúdo :

Javascript-escritor

Copiar código

```
elemento.innerHTML = 'Novo conteúdo aqui!';
```

2.

Alterar atributos :

Javascript

Copiar código

```
elemento.setAttribute('class', 'novaClasse');
```

3.

Adicionar novos elementos :

JavaS

Copiar código

```
let novoElemento = document.createElement('p');  
novoElemento.  
novoElemento  
innerHTML = 'Este é um novo parágrafo!';  
  
docum  
document.body.appendChild(novoElemento);
```

4.

Ouvir eventos :

javascript

Copiar código

```
let botao = document.getElementById('meuBotao');  
botao.addEventListener('click', function() {  
  
  
alert('Botão clicado!');  
});
```

5.

Em resumo, o **DOM** é essencial

4o mini

Dito por si:

oq e dom em javascript

]~`jnoñipj]

pO ****DOM**** (Document Object Model) em JavaScript é uma interface de programação que permite que scripts (como o JavaScript) acessem e modifiquem

o conteúdo, a estrutura e o estilo de um documento HTML ou XML. Em resumo: - O DOM é uma representação em forma de **árvore** do documento HTML ou XML. Cada parte do documento (elementos, atributos, textos) é representada por **nós** na árvore. - Com o DOM, o JavaScript pode acessar esses **nós** e manipulá-los, alterando o conteúdo da página, sua estrutura e comportamento. **### Estrutura do DOM:** Imagine um documento HTML como uma árvore, onde cada elemento HTML é um nó da árvore: `<html> <head> <title>Exemplo DOM</title> </head> <body> <h1>Olá, Mundo!</h1> <p>Este é um parágrafo.</p> </body> </html>` Essa estrutura seria representada no DOM como uma árvore de objetos, onde: - O nó raiz é o `<html>`. - O nó filho do `<html>` seria o `<head>` e o `<body>`. - Dentro do `<body>`, haveria nós para `<h1>` e `<p>`, e assim por diante. **### Como a interação JavaScript com o DOM?** JavaScript usa o DOM para manipular uma página. Você pode, por exemplo: - **Alterar conteúdo do texto**. - **Adicionar ou remover elementos**. - **Alterar atributos** (como classes, ids, estilos). - **Reagir a eventos** (como cliques, pressionamento de teclas, etc.). **### Exemplos de uso do DOM em JavaScript:** 1. **Selecionando um elemento**: `javascript const titulo = document.querySelector('h1');` // Seleciona o primeiro `<h1>` na página 2. *

4o mini

Como Funciona

1. HTML :

- Temos um campo de entrada (`<input>`) onde `<button>` `` sobre

2. CSS :

- Estilo

3. JavaScript :

- Seleccionamos os elementos do DOM usando `document.getElementById()`.
- Uma articulação `adicionarTarefa()` criar ``, anúncio
- Usamos `addEventListener` para
 - Ao clicar em um item da lista (``), ele
 - O
 - Tamb

Funções

- **Adicionar tarefas :**
- **Marcar tarefas como concluídas :** Ao
- **Remover tarefas :** A

Resumo

Essa aplicação permite que você interaja com uma página de formato dinâmico, usando as funcionalidades do DOM para adicionar, remover e modificar elementos da página, tudo com JavaScript. Isso é um exemplo clássico de como o DOM pode ser utilizado para criar

4o mini

Dito por si:

aplicação de dom em javascript em trexo menor

Dito pelo ChatGPT:

ChatGPT

Aqui está um exemplo mais compacto de uma **aplicação de DOM em JavaScript** , o

Exemplos

HTML

altura

Copiar código

```
<!DOCTYPE html>

<html lang="pt-br">

<head>
    <meta charset="UTF-8">


<title>Lista de Tarefas</title>

</h1>
</head>
<body>
    <h1>Lista de Tarefas</h1>


    <input type="text" id="tarefaInput" placeholder="Nova tarefa">


    <button id="addTarefaBtn">Adicionar</button>


    <ul id="listaTarefas"></ul>
```

Exemplos práticos de manipulação do DOM com JavaScript

O DOM (Document Object Model) é uma interface poderosa que permite ao JavaScript interagir com os elementos HTML de uma página web. Essa interação possibilita a criação de páginas dinâmicas e interativas.

Para ajudá-lo a entender melhor, vamos explorar alguns exemplos:

1. Alterando o Conteúdo de um Elemento:

JavaScript

```
// Selecionando o elemento com o ID "meuParagrafo"
const paragrafo = document.getElementById("meuParagrafo");

// Alterando o texto do parágrafo
paragrafo.textContent = "Novo texto inserido pelo JavaScript!";
```

2. Alterando o Estilo de um Elemento:

JavaScript

```
// Selecionando o elemento com a classe "caixa"
const caixa = document.querySelector(".caixa");

// Alterando a cor de fundo e a largura
caixa.style.backgroundColor = "blue";
caixa.style.width = "200px";
```

3. Criando um Novo Elemento:

JavaScript

```
// Criando um novo elemento <p>
const novoParagrafo = document.createElement("p");

// Adicionando texto ao novo parágrafo
novoParagrafo.textContent = "Este parágrafo foi criado dinamicamente!";

// Selecionando um elemento pai para adicionar o novo parágrafo
const container = document.getElementById("container");

// Adicionando o novo parágrafo ao container
container.appendChild(novoParagrafo);
```

4. Adicionando um Evento:

JavaScript

```
// Selecionando um botão
```



```
const botao = document.getElementById("meuBotao");

// Adicionando um evento de clique
botao.addEventListener("click", function() {
    alert("Você clicou no botão!");
});
```

5. Removendo um Elemento:

JavaScript

```
// Selecionando o elemento a ser removido
const elementoParaRemover = document.getElementById("elementoParaRemover");

// Removendo o elemento do DOM
elementoParaRemover.parentNode.removeChild(elementoParaRemover);
```

6. Alternando uma classe:

JavaScript

```
// Select the element

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

```
// Function to toggle the "hidden" class
```

```
function toggleHidden() {
```

```
    element.classList.toggle("hidden");
```

```
}
```

```
// Add a click event listener to a button to trigger the toggle
```

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

```
button.addEventListener("click", toggleHidden);
```

- **Criando elementos de lista:** Adicionando itens a uma lista não ordenada ou ordenada.
- **Ocultando e mostrando elementos:** Alterando a propriedade display de um elemento.
- **Validando formulários:** Verificando se os campos de um formulário estão preenchidos corretamente.
- **Criando efeitos de animação:** Utilizando JavaScript e CSS para criar animações suaves.
- **Manipulando tabelas:** Adicionando, removendo e modificando linhas e colunas de uma tabela.

Conceitos-chave do DOM:

- **Nós:** Representamos os elementos HTML, atributos e texto do documento.
- **Árvore DOM:** Uma estrutura hierárquica que representa o documento.

- **Métodos e propriedades:** Permitem manipular os nós da árvore DOM.

Recursos Adicionais:

- **MDN Web Docs:** A documentação oficial da Mozilla Developer Network é um excelente recurso para aprender mais sobre o DOM.
- **W3Schools:** Oferece tutoriais e exemplos práticos sobre o DOM.

Com o DOM, você pode criar páginas web que respondam às ações do usuário, atualizem o conteúdo dinamicamente e ofereçam uma experiência mais rica e interativa.

Gostaria de explorar alguns desses exemplos com mais detalhes ou ver outros exemplos?

Você tem alguma dúvida específica sobre o DOM?

No entanto, podemos destacar 6 ações comuns que você realiza ao manipular o DOM:

1. Selecionando elementos:

- **getElementById:** Seleciona um elemento pelo seu ID.
- **getElementsByClassName:** Seleciona elementos por suas classes.

- **getElementsByTagName:** Seleciona elementos por suas tags.
- **querySelector:** Seleciona o primeiro elemento que corresponde a um seletor CSS.
- **querySelectorAll:** seleciona todos os elementos que dependem de um seletor CSS.

2. Modificando o conteúdo:

- **textContent:** altera o texto dentro de um elemento.
- **innerHTML:** altera o HTML dentro de um elemento.

3. Modificando estilos:

- **style:** Altera o estilo inline de um elemento.
- **classList:** Adiciona, remove ou verifica classes de um elemento.

4. Criando novos elementos:

- **createElement:** Cria um novo elemento.
- **appendChild:** Adiciona um novo elemento como filho de outro.

5. Removendo elementos:

- **removeChild:** Remove um elemento filho.

6. Adicionando eventos:

- **addEventListener:** Adiciona um ouvinte de evento a um elemento.

CONCLUSÃO:

O DOM em JavaScript é uma interface de programação que permite a interação com documentos HTML e XML. Ele representa a estrutura de uma página web como uma árvore de objetos, onde cada elemento HTML é um nó que pode ser manipulado via JavaScript. Com o DOM, é possível alterar dinamicamente o conteúdo e o estilo da página, responder a eventos do usuário, como cliques e teclas pressionadas, e até mesmo criar novos elementos. Ele é fundamental para o desenvolvimento de páginas interativas e dinâmicas, proporcionando uma experiência de usuário mais rica. Além disso, o DOM facilita a manipulação de dados e a atualização de conteúdos sem a necessidade de recarregar a página inteira.

- **Criando elementos de lista:** Adicionando itens a uma lista não ordenada ou ordenada.
- **Ocultando e mostrando elementos:** Alterando a propriedade display de um elemento.
- **Validando formulários:** Verificando se os campos de um formulário estão preenchidos corretamente.
- **Criando efeitos de animação:** Utilizando JavaScript e CSS para criar animações suaves.
- **Manipulando tabelas:** Adicionando, removendo e modificando linhas e colunas de uma tabela.

Conceitos-chave do DOM:

- **Nós:** Representamos os elementos HTML, atributos e texto do documento.

- **Árvore DOM:** Uma estrutura hierárquica que representa o documento.
- **Métodos e propriedades:** Permitem manipular os nós da árvore DOM.

Recursos Adicionais:

- **MDN Web Docs:** A documentação oficial da Mozilla Developer Network é um excelente recurso para aprender mais sobre o DOM.
- **W3Schools:** Oferece tutoriais e exemplos práticos sobre o DOM.

Com o DOM, você pode criar páginas web que respondam às ações do usuário, atualizem o conteúdo dinamicamente e ofereçam uma experiência mais rica e interativa.

Gostaria de explorar alguns desses exemplos com mais detalhes ou ver outros exemplos?

Você tem alguma dúvida específica sobre o DOM?

remov

```
textContent = 'Remover';  
    removerBtn.  
    removerB
```

```
onclick = () => li.remove(); // Remove a tarefa
```

```
    li.
```

```
appendChild(removerBtn);
```

```
    docume
```

```
document.getElementById('listaTarefas').appendChild(li);
```

```
    doc
```

```
document.getElementById('tarefaInput').value = ''; // Limpa o campo de input  
});
```

C

1. HTML :

- Tem um campo de entrada (`<input>) p<button>) p

2. JavaScript :

- Quando o botão "Adicionar" é clicado, o texto do campo
- Um novo item da lista (``

```
<script src="script.js"></script>
```

```
</body>
```

```
</h
```

```
</html>
```

JavaScript (arquivo `script.js`):

japão

Copiar código

```
document.getElementById('addTarefaBtn').addEventListener('click', function()  
{
```

```
co
```

```
const tarefaTexto = document.getElementById('tarefaInput').value;
```

```
i
```

```
if (tarefaTexto.trim() === '') return; // Verifica se não está vazio
```

```
const li = document.createElement('li');
```

```
li.
```

```
l
```

```
textContent = tarefaTexto;
```

```
const removerBtn = document.createElement('button');
```