

Linguagem de Programação



Prof. Renato Carioca Duarte

Unidade 04

Interface com Usuário

Documento Object Model

- O **Document Object Model** ou simplesmente DOM é utilizado pelo navegador Web para representar a sua página Web.
- Quando altera-se esse modelo com o uso do Javascript altera-se também a página Web.
- É muito mais fácil trabalhar com DOM do que diretamente com código HTML ou CSS.

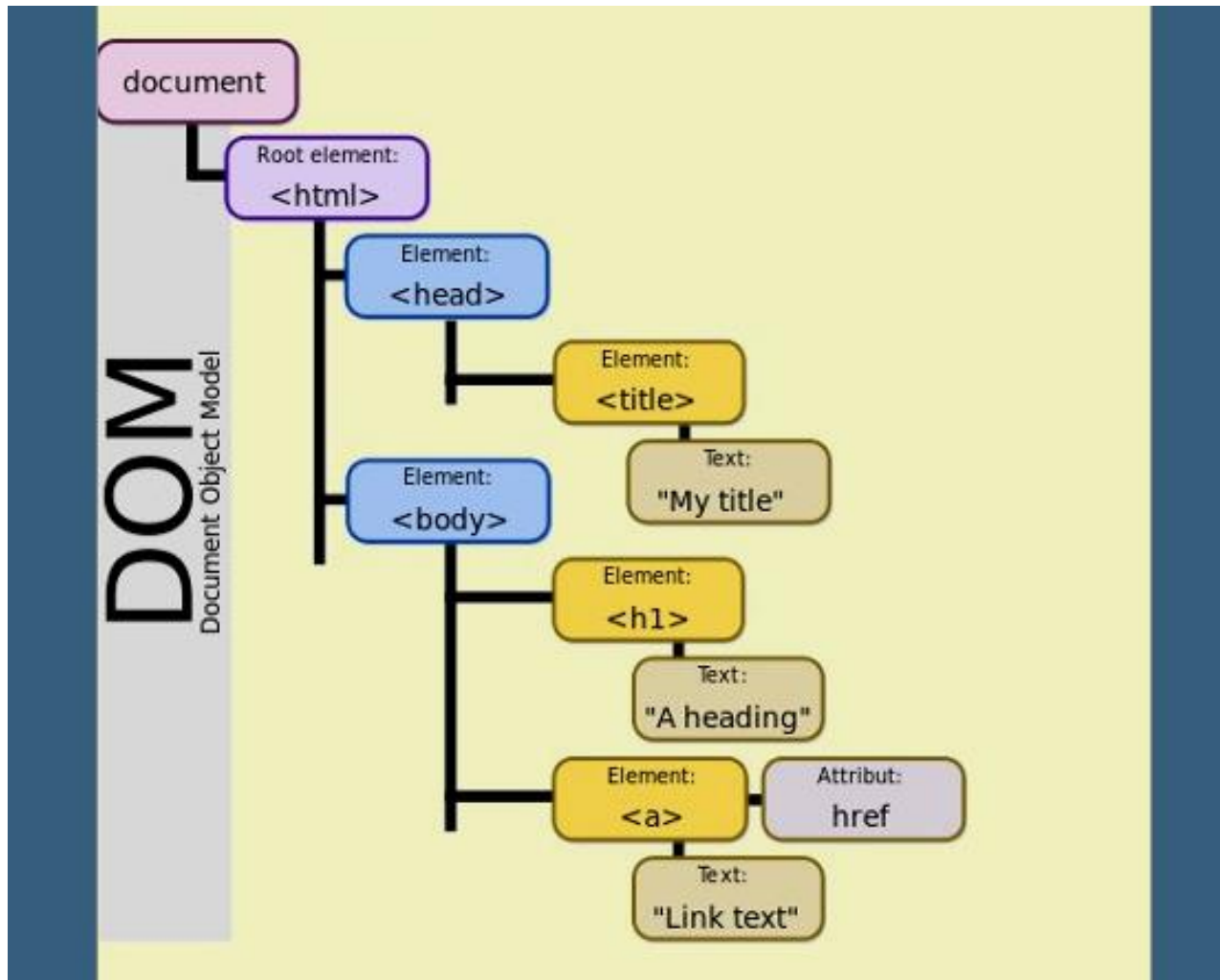
Documento Object Model

- Um dos grandes responsáveis por isso tudo é o **objeto document** que é responsável por conceder ao código Javascript todo o acesso a árvore DOM do navegador Web.
- Portanto, qualquer coisa criado pelo navegador Web no modelo da página Web poderá ser acessado através do objeto Javascript document.

Documento Object Model

- O DOM (Documento Object Model) é uma interface de programação fornecida pelos navegadores web que representa a estrutura HTML e XML de uma página como uma árvore de objetos.
- Ele permite que os programas em JavaScript interajam e manipulem os elementos, conteúdos e estilos da página de forma dinâmica, tornando possível a criação de páginas web interativas e dinâmicas.

Document Object Model



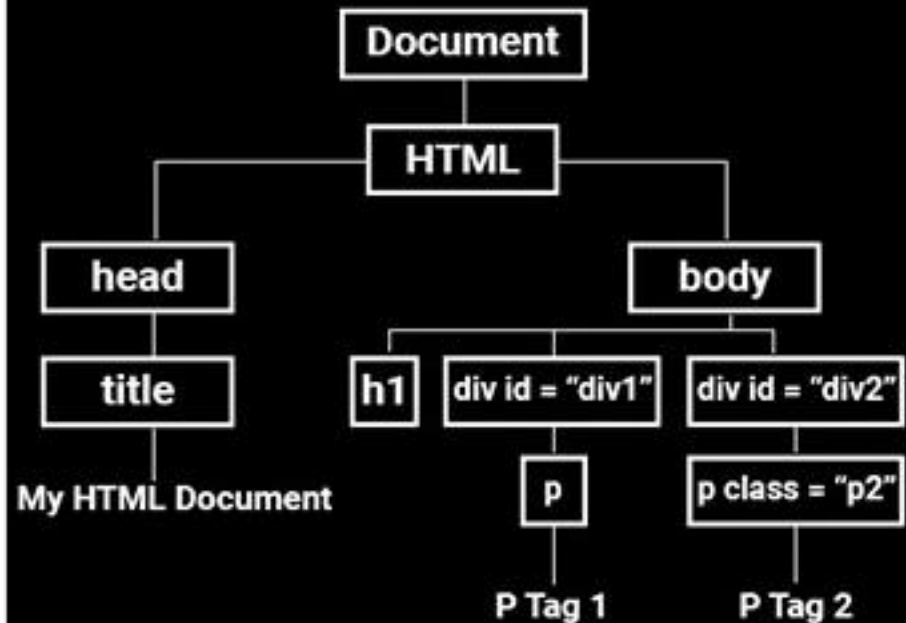
Documento Object Model

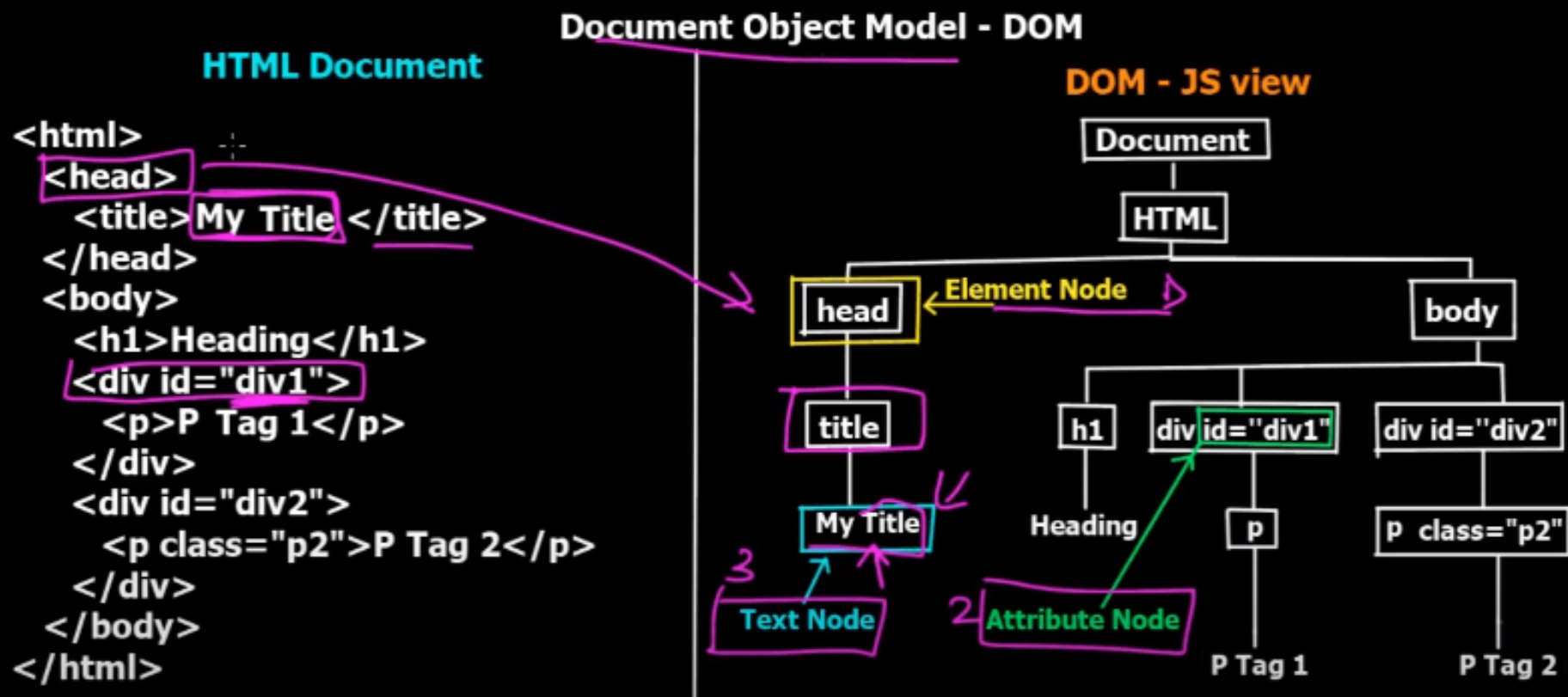
- O DOM organiza a página web em uma estrutura hierárquica, onde cada elemento HTML é representado por um objeto no DOM.
- Os objetos do DOM podem ser acessados e manipulados usando JavaScript, permitindo a adição, remoção e modificação dos elementos da página, bem como a interação com eventos do usuário.

HTML Document

```
index.html x
1  <html>
2    <head>
3      <title>My HTML Document</title>
4    </head>
5
6    <body>
7      <h1>Heading</h1>
8      <div id="div1">
9        <p>P Tag 1</p>
10     </div>
11     <div id="div2">
12       <p class="p2">P Tag 2</p>
13     </div>
14   </body>
15 </html>
```

Document Object Model (DOM)





Manipulação do DOM

- A **manipulação do DOM** (Document Object Model) com JavaScript envolve interações com a estrutura HTML e a página da web.
- O DOM é uma representação em árvore dos elementos HTML que compõem a página, e JavaScript permite acessar, modificar e interagir com esses elementos de forma dinâmica.

1. Seleção de elementos:

Selecionar elementos HTML no DOM usando métodos como ``getElementById``, ``getElementsByClassName``, ``getElementsByTagName``, ``querySelector``, ``querySelectorAll``, entre outros.

2. Manipulação de conteúdo:

Alterar ou obter o conteúdo dos elementos HTML, seja o texto ou o HTML interno, usando as propriedades ``innerText``, ``innerHTML``, ``textContent``, ``value``, etc.

3. Manipulação de atributos:

Adicionar, modificar ou remover atributos dos elementos HTML usando as propriedades ``setAttribute``, ``getAttribute``, ``removeAttribute``, etc.

4. Estilização:

Alterar os estilos de elementos HTML usando as propriedades ``style`` ou adicionar e remover classes para modificar a aparência de elementos.

5. Eventos:

Lidar com eventos do DOM, como cliques de mouse, pressionamento de teclas, envio de formulários, entre outros. Isso envolve a adição de event listeners usando métodos como ``addEventListener``.

6. Criação e manipulação de elementos:

Criar novos elementos HTML dinamicamente, adicionar e remover elementos no DOM, bem como mover elementos de um local para outro.

7. Traversing (navegação) do DOM:

Percorrer a estrutura do DOM para encontrar elementos pai, filhos ou irmãos usando as propriedades como ``parentNode``, ``childNodes``, ``nextSibling``, ``previousSibling``, etc.

8. Animações e Transições:

Usar JavaScript para criar animações e transições, modificando propriedades CSS ao longo do tempo para criar efeitos visuais.

9. Formulários:

Interagir com elementos de formulário, como capturar valores, validar dados, enviar formulários e manipular eventos relacionados aos elementos de formulário.

Manipulação de Eventos

- A manipulação de eventos é um dos conceitos centrais em JavaScript para desenvolvimento web, permitindo a interação entre o usuário e a página web.
- Basicamente, um evento é qualquer coisa que acontece na página, como um clique do mouse, pressionar uma tecla, mover o mouse, etc.
- A manipulação de eventos refere-se ao processo de captura e resposta a esses eventos.

Manipulação de Eventos

1. Event Listeners:

Um event listener é uma função que é registrada para ser chamada quando um determinado evento ocorre em um elemento específico.

Ele "escuta" o evento e responde a ele executando o código dentro da função.

2. Tipos de Eventos:

Existem muitos tipos de eventos que você pode capturar, como cliques (`click`), pressionamentos de tecla (`keydown`, `keyup`, `keypress`), foco (`focus`, `blur`), movimento do mouse (`mousemove`, `mouseover`, `mouseout`), envio de formulários (`submit`), entre outros.

3. Objeto Event:

Quando um evento ocorre, um objeto chamado `Event` é criado automaticamente.

Esse objeto contém informações sobre o evento, como o tipo de evento, o elemento alvo (onde o evento ocorreu), coordenadas do mouse, teclas pressionadas, entre outros detalhes úteis.

4. Prevenção de Comportamentos Padrão:

Em alguns casos, é necessário impedir que o comportamento padrão do navegador ocorra em resposta a um evento específico.

Por exemplo, impedir que um formulário seja enviado quando o usuário pressiona a tecla "Enter" dentro de um campo de texto.

5. Delegação de Eventos:

Quando você tem muitos elementos semelhantes na página (por exemplo, itens em uma lista), pode ser mais eficiente adicionar um único event listener no elemento pai (como uma lista) em vez de adicionar um event listener para cada elemento individual.

Essa técnica é chamada de delegação de eventos.

Manipulação de Eventos

Exemplo de código em HTML e JavaScript que exemplifica o DOM:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Exemplo do DOM</title>
  </head>
  <body>
    <h1 id="titulo">Título da Página</h1>
    <p id="paragrafo">Este é um parágrafo.</p>
    <button id="botao">Clique aqui</button>

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

```
// Selecionando elementos do DOM pelo id
const tituloElemento = document.getElementById("titulo");
const paragrafoElemento = document.getElementById("paragrafo");
const botaoElemento = document.getElementById("botao");

// Alterando o conteúdo e estilo dos elementos
tituloElemento.innerText = "Novo Título";
paragrafoElemento.innerHTML =
  "Este é um parágrafo modificado. <strong>Em negrito!</strong>";
botaoElemento.style.backgroundColor = "blue";
botaoElemento.style.color = "white";

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

Manipulação de Eventos

- Neste exemplo, temos um código HTML simples com um título, um parágrafo e um botão.
- O arquivo script.js contém o código JavaScript que manipula o DOM da página.
- Usamos o método `document.getElementById()` para selecionar elementos do DOM pelo id e atribuímos esses elementos a variáveis (tituloElemento, paragrafoElemento e botaoElemento).
- Em seguida, usamos as propriedades e métodos do DOM para alterar o conteúdo e o estilo desses elementos.

Manipulação de Eventos

- Por exemplo, alteramos o texto do título usando ``innerText``, o conteúdo do parágrafo usando ``innerHTML`` e alteramos o estilo do botão usando as propriedades ``style.backgroundColor`` e ``style.color``.
- Além disso, adicionamos um evento de clique ao botão usando o método ``addEventListener``.
- Quando o botão é clicado, uma mensagem de alerta é exibida.

Manipulação de Formulários

- A manipulação de formulários e a validação de entrada são tarefas essenciais ao trabalhar com páginas web que contêm formulários interativos.
- Os formulários são uma das principais formas de interação do usuário com o conteúdo da página e permitem que os usuários enviem dados para o servidor.
- A manipulação e validação desses formulários são importantes para garantir que os dados sejam submetidos corretamente e que sejam confiáveis para serem processados.
- Aqui estão os principais conceitos relacionados à manipulação de formulários e validação de entrada em JavaScript:

1. Seleção de elementos do formulário:

Use os métodos ``getElementById``, ``querySelector``, etc., para selecionar elementos do formulário e acessar seus valores.

2. Eventos de formulário:

Use event listeners para capturar eventos de formulário, como o envio do formulário (``submit``), mudanças em campos de entrada (``change``, ``input``), entre outros.

3. Prevenção do envio padrão:

Ao manipular o evento de envio do formulário (``submit``), você pode usar ``event.preventDefault()`` para evitar que o formulário seja submetido automaticamente, permitindo que você valide os dados e tome ações personalizadas.

4. Validação de entrada:

Verifique se os dados inseridos nos campos de entrada são válidos antes de enviar o formulário. Você pode usar expressões regulares (``RegExp``), validação numérica ou qualquer outra lógica personalizada para garantir que os dados estejam no formato correto.

5. Feedback ao usuário:

Forneça feedback claro e útil ao usuário, indicando quais campos precisam ser corrigidos caso haja erro de validação.

6. Exibição de mensagens de erro:

Ao validar a entrada do usuário, você pode exibir mensagens de erro ou realçar os campos com problemas para orientar o usuário sobre como corrigir a entrada incorreta.

7. Envio de dados:

Depois de validar os dados do formulário, você pode enviá-los ao servidor usando a submissão do formulário ou por meio de requisições HTTP, conforme necessário.

validação de entrada em um formulário

```
<form id="meuFormulario">  
  <label for="email">E-mail:</label>  
  <input type="email" id="email" required />  
  <br />  
  <label for="senha">Senha:</label>  
  <input type="password" id="senha" required />  
  <br />  
  <button type="submit">Enviar</button>  
</form>
```

validação de entrada em um formulário

```
const formulario = document.getElementById("meuFormulario");

formulario.addEventListener("submit", function (event) {
  const emailInput = document.getElementById("email");
  const senhaInput = document.getElementById("senha");

  if (!emailInput.value || !senhaInput.value) {
    event.preventDefault(); // Evita o envio do formulário
    alert("Por favor, preencha todos os campos.");
  }
});
```


validação de entrada em um formulário

- Neste exemplo, o evento submit é adicionado ao formulário usando o método `addEventListener`.
- Quando o formulário é enviado, a função anônima (callback) é executada.
- Dentro do callback, recuperamos os valores dos campos de e-mail e senha usando `getElementById` e verificamos se ambos estão preenchidos (`!emailInput.value || !senhaInput.value`).
- Se algum dos campos não estiver preenchido, o evento submit é impedido de prosseguir (usando `event.preventDefault()`) e um alerta é exibido pedindo para preencher todos os campos.

validação de entrada em um formulário

- Essa é uma validação básica e pode ser estendida para incluir outras verificações mais complexas, como verificar se o campo de e-mail contém um endereço de e-mail válido ou se a senha atende a determinados requisitos de complexidade.
- A validação no lado do cliente é uma boa prática para fornecer uma experiência melhor para os usuários e evitar envio de dados inválidos ao servidor.



JS

Dúvidas

??????



Questionário

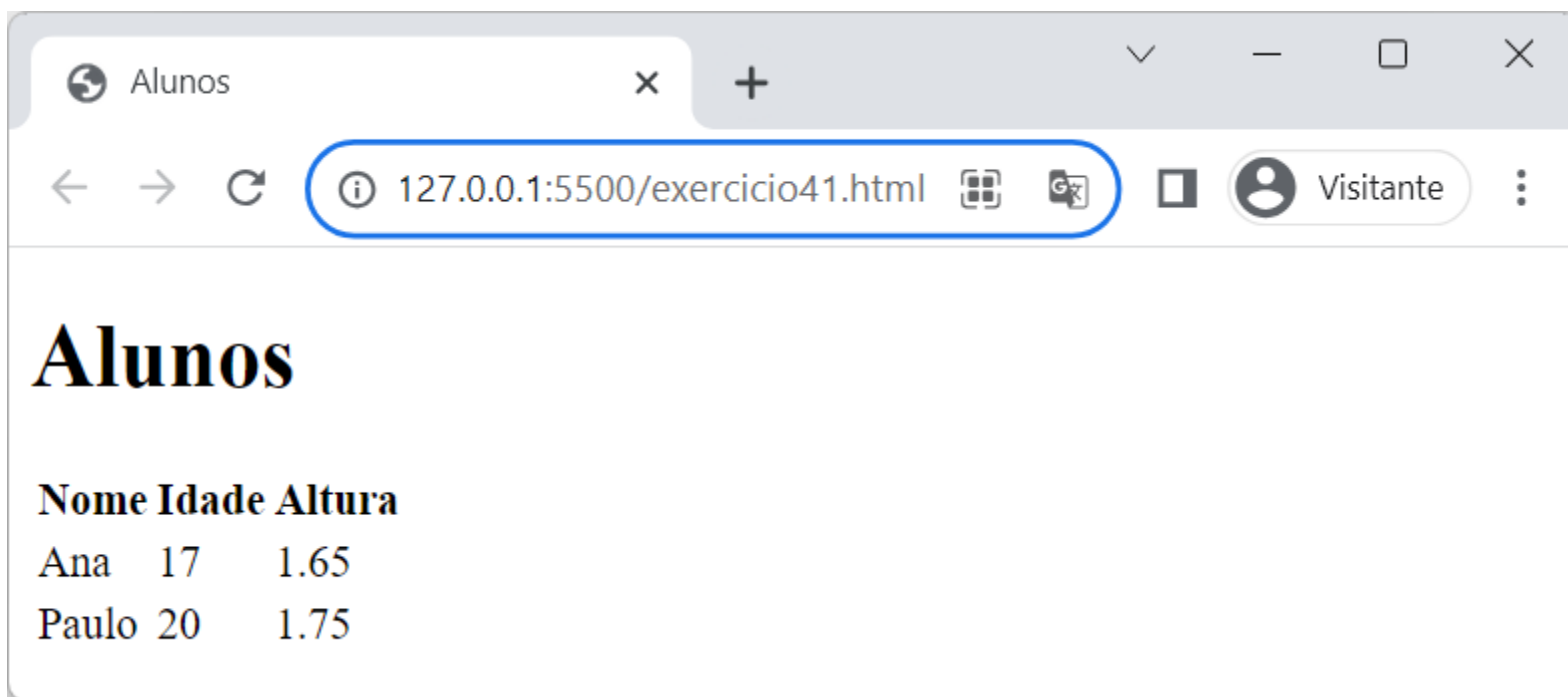


Questionário Unidade 4 no classroom

1. Crie uma tabela no HTML contendo os cabeçalhos Nome, Idade e Altura de Alunos.

Para preencher os dados, crie um script que coloque os alunos "Ana", 17 anos, 1.65 e "Paulo", 20 anos e 1.75 em um array e use a o método `forEach` para preencher a tabela.

```
<table>
  <thead>
    <tr>
      <th>Nome</th>
      <th>Idade</th>
      <th>Altura</th>
    </tr>
  </thead>
  <tbody>
    <!-- Os dados serão preenchidos aqui -->
  </tbody>
</table>
```



The screenshot shows a web browser window with a single tab titled 'Alunos'. The address bar displays the URL '127.0.0.1:5500/exercicio41.html' and indicates the user is a 'Visitante'. The page content features a large heading 'Alunos' and a table with three columns: 'Nome', 'Idade', and 'Altura'. The table contains two rows of data: 'Ana' (17, 1.65) and 'Paulo' (20, 1.75).

Nome	Idade	Altura
Ana	17	1.65
Paulo	20	1.75

2. Crie um formulário HTML que tenha campos de input para nome, e-mail e senha.

Faça um script JS para validar que nenhum campo esteja vazio antes de submeter.



The screenshot shows a web browser window with the title "Formulário de Validação". The address bar displays the URL "127.0.0.1:5500/exercicio42.ht...". The page content includes the heading "Cadastro de Cliente" and three input fields labeled "Nome:", "E-mail:", and "Senha:". Below these fields is a "Submeter" button.

Cadastro de Cliente

Nome:

E-mail:

Senha:

Se deixar algum campo em branco:

