

A woman with long dark hair, wearing glasses, is shown from the chest up, looking down at a laptop screen. The background is dark and slightly blurred.

IN

INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

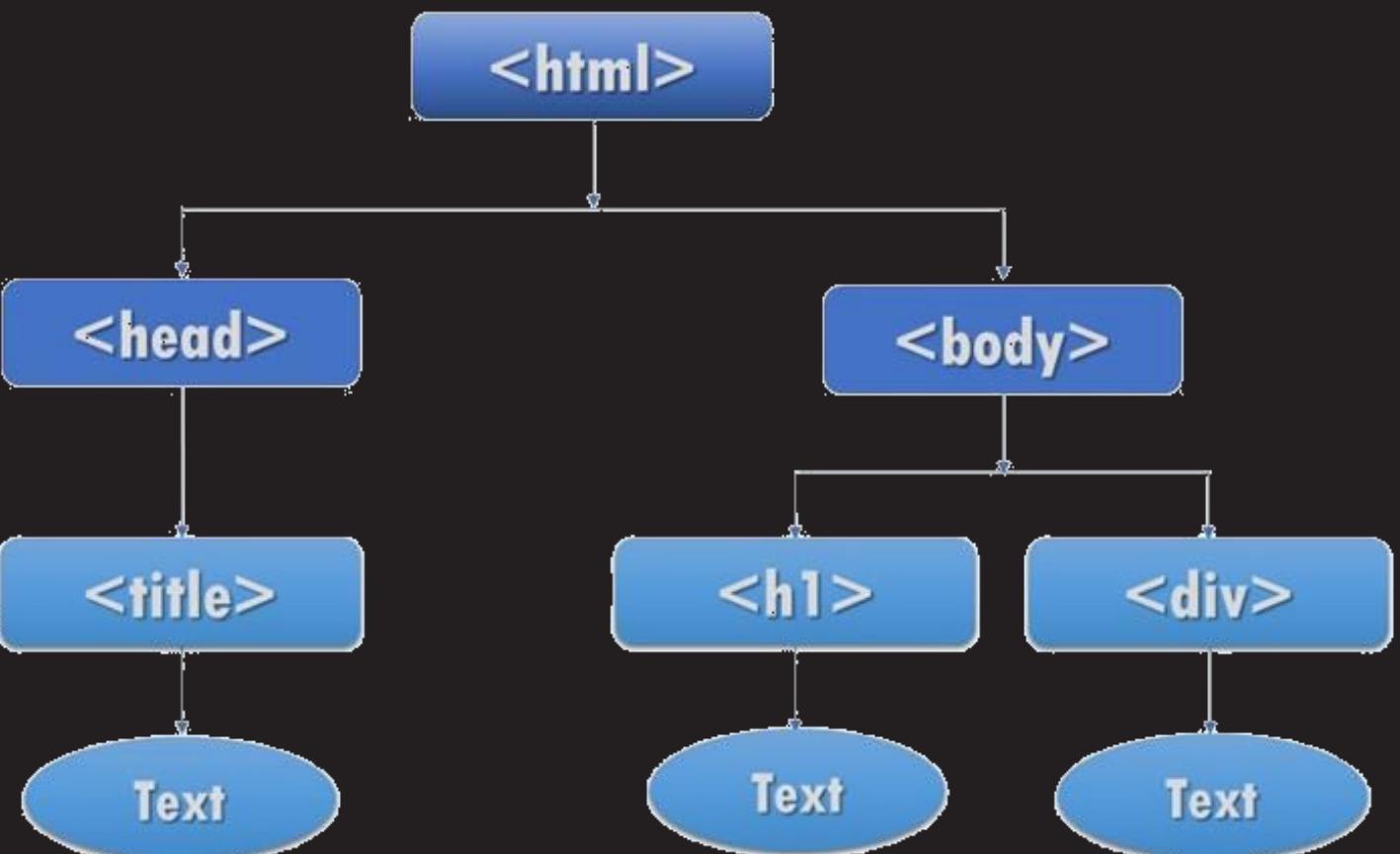
AULA 09 - EVENTOS DOM II

O QUE IREMOS APRENDER

- 01** RESUMO DA AULA PASSADA
- 02** EVENTOS DOM
- 03** TIPOS DE EVENTOS
- 04** CRIAR ELEMENTOS
- 05** ESTILIZANDO ELEMENTOS PELO DOM
- 06** MÃOS AO CÓDIGO

RESUMO DA AULA PASSADA

DOM (Document Object Model) é uma representação da nossa árvore de elementos do HTML. Ela nos traz todo o modelo do nosso documento HTML, nos permitindo manipulá-lo de diversas formas. Ela se assemelha muito a uma árvore genealógica, onde cada elemento tem seus respectivos pai e filhos.



EVENTOS DOM

Os eventos DOM permitem que você crie interatividade e resposta do usuário em suas páginas da web. Quando um evento ocorre, o navegador dispara o evento correspondente e chama uma função de tratamento (callback) associada a esse evento.

Essa função de tratamento é executada para lidar com a ação ou alteração específica que ocorreu.

EVENTOS DOM

```
<body>
  <button id="meuBotao">Clique-me!</button>
  <p id="mensagem"></p>
  <script>
    // Obtém referências aos elementos de botão e parágrafo no HTML
    const botao = document.getElementById('meuBotao');
    const mensagemParagrafo = document.getElementById('mensagem');

    // Registra um evento de clique no botão
    botao.addEventListener('click', () => {
      mensagemParagrafo.innerHTML = 'Olá Raama';
    });
  </script>
</body>
```

EVENTOS DOM

addEventListener e **onclick** são duas maneiras diferentes de associar funções de tratamento de eventos a elementos HTML para responder a ações do usuário. Ambos são usados para lidar com eventos DOM, mas existem algumas diferenças entre eles:



EVENTOS DOM

`addEventListener`: É considerado uma prática recomendada, pois separa o código JavaScript do HTML, promovendo uma melhor organização e manutenção do código. É mais flexível, pois permite adicionar múltiplos ouvintes para o mesmo tipo de evento em um elemento, bem como remover ouvintes individualmente.

```
<script>
  const botao = document.getElementById('meuBotao');

  botao.addEventListener('click', () => {
    console.log('Botão clicado usando addEventListener');
  });
</script>
```

EVENTOS DOM

onclick: Mistura o código JavaScript com o HTML, o que pode tornar o código menos organizado e mais difícil de manter, especialmente em projetos maiores. É menos flexível, pois você só pode associar uma única função de tratamento de evento ao elemento.

```
<body>
  <button id="meuBotao" onclick="console.log('Botão clicado usando onclick')">Clique-me!</button>
</body>
```

TIPOS DE EVENTOS

Diversos exemplos de tipos de eventos estão disponíveis para utilização com o método addEventListener. Cada tipo de evento está vinculado a um comportamento particular e pode ser aproveitado para a criação de interações envolventes e dinâmicas em suas páginas da web. A seguir, serão apresentados alguns exemplos ilustrativos de tipos de eventos. No entanto, para obter informações mais abrangentes, você pode realizar pesquisas adicionais explorando recursos disponíveis, como o seguinte link:

https://www.w3schools.com/jsref/dom_obj_event.asp

TIPOS DE EVENTOS

input: Acionado quando o valor de um campo de entrada muda.

```
<body>
  <input type="text" id="meuInput">
  <script>
    const input = document.getElementById('meuInput');
    input.addEventListener('input', (event) => {
      console.log(`Texto digitado: ${event.target.value}`);
    });
  </script>
</body>
```

TIPOS DE EVENTOS

mouseover: Ele é acionado quando o cursor do mouse entra em cima de um elemento HTML.

```
<body>
  <div id="minhaDiv">AQUI ESTÁ MINHA DIV</div>
  <script>
    const div = document.getElementById('minhaDiv');
    div.addEventListener('mouseover', () => {
      div.innerHTML = 'Olá Raama'
    });
  </script>
</body>
```

TIPOS DE EVENTOS

mouseout: Ele é acionado quando o cursor do mouse sai em cima de um elemento HTML.

```
<body>
  <div id="minhaDiv">AQUI ESTÁ MINHA DIV</div>
  <script>
    const div = document.getElementById('minhaDiv');
    div.addEventListener('mouseout', () => {
      div.innerHTML = 'Olá Raama'
    });
  </script>
</body>
```

CRIAR ELEMENTOS

Pelo DOM é possível criar elementos e configurá-los dinamicamente em resposta a eventos usando JavaScript. Os eventos DOM permitem que você responda a interações do usuário e, ao lidar com esses eventos, você pode criar e modificar elementos HTML em tempo real.

CRIAR ELEMENTOS

```
<body>
  <!-- Elementos existentes na página -->
  <div id="conteudo"> </div>
  <script>
    // Criar um novo elemento div
    const novoDiv = document.createElement('div');

    // Configurar o elemento criado com algumas propriedades
    novoDiv.textContent = 'Este é um novo elemento div.';
    novoDiv.style.backgroundColor = 'lightblue';
    novoDiv.style.padding = '10px';
    novoDiv.style.border = '1px solid blue';

    // Adicionar o novo elemento à página, dentro do elemento com o ID "conteudo"
    const conteudo = document.getElementById('conteudo');
    conteudo.appendChild(novoDiv);
  </script>
</body>
```

CRIAR ELEMENTOS

A função `createElement()` irá criar um elemento HTML para ser, posteriormente, inserido em um documento HTML.

```
<script>
  let element = document.createElement(tagName);
  // tagName é uma string que especifica o tipo do elemento, é uma tag HTML.
  // element é o objeto criado e retornado pela função

</script>
```

CRIAR ELEMENTOS

Ao invés de criar e anexarmos um nó de texto podemos lançar mão da propriedade `textContent` do elemento que estamos criando.

```
<script>
    // Criar elemento
    let titulo = document.createElement('h1');
    // Criando o nó de texto de outra forma
    titulo.textContent = "Um título qualquer"

</script>
```

CRIAR ELEMENTOS

Para estilizar um elemento pelo DOM (Document Object Model) em JavaScript, você pode usar a propriedade `style` do objeto do elemento para modificar suas propriedades de estilo. Essa é uma maneira simples de aplicar estilos diretamente em elementos individuais usando JavaScript.

```
elemento.style.nomeDaPropriedade = 'valor';
```

CRIAR ELEMENTOS

```
<body>
  <button onclick="criar()">Criar Quadrado</button>
  <div class="box"></div>

  <script>
function criar(){
  let box = document.querySelector('.box');
  box.style.width = '100px';
  box.style.height = '100px';
  box.style.background = '#f00';
  box.style.color = '#f00';
}
  </script>
</body>
```

CRIAR ELEMENTOS

O exemplo ao lado estiliza o parágrafo, de acordo com a cor inserida no input pelo usuário.

```
<body>
  <div>
    <input type="text" id="corInput" placeholder="Digite a cor " >
    <button id="estilizarBotao">Estilizar Parágrafo</button>
  </div>

  <p id="paragrafo">Este é um parágrafo de exemplo.</p>

  <script>
    // Função para estilizar o parágrafo com a cor fornecida
    function estilizarParagrafo() {
      const cor = document.getElementById('corInput').value;

      const paragrafo = document.getElementById('paragrafo');
      paragrafo.style.color = cor;

      cor.value = ''; // Limpar o campo de entrada após estilizar
    }

    // Associar o evento de clique ao botão "Estilizar Parágrafo"
    const estilizarBotao = document.getElementById('estilizarBotao');
    estilizarBotao.addEventListener('click', estilizarParagrafo);

  </script>
</body>
</html>
```



ATIVIDADE PRÁTICA

Atividade 01

Crie uma caixa de entrada de texto e um botão. Quando o botão é clicado, criar uma lista não ordenada (``) e adicionar cada palavra digitada na caixa de entrada como um novo item da lista (``).

Atividade 02

Crie um botão que, quando clicado, adiciona um novo parágrafo com um texto personalizado.

ATIVIDADE PRÁTICA

Atividade 03

Crie uma paleta de cores usando botões coloridos. Quando um botão de cor é clicado, definir a cor de fundo do corpo da página para essa cor selecionada.

Atividade 04

Crie um gerador de citações que exibirá citações aleatórias na página. Cada vez que o usuário clicar em um botão "Gerar Citação", uma citação aleatória será exibida, e o estilo da citação mudará dinamicamente.

ATIVIDADE PRÁTICA

Atividade 05

Aprofunde seu conhecimento após dominar o evento 'click', explorando agora o 'mouseover': pratique alternando a imagem 'lampada.jpg' para 'lampada-on.jpg' sempre que o mouse for posicionado sobre ela, conforme o exemplo ilustrado abaixo.

ATIVIDADE PRÁTICA

Atividade 06

Crie um evento que receba os valores que foram digitados nos campos “inputValorA” e “inputValorB”. Para isto, você deverá criar uma variável chamada botaoCalculo e utilizar o document.getElementById para selecionar o id do botão(“calculo”).

ATIVIDADE PRÁTICA

Atividade 07

Desenvolva uma interação dinâmica usando o evento 'mouseover'. Crie uma div em seu arquivo HTML e, quando o cursor do mouse for movido sobre ela, o background da div deve mudar de cor. Ao retirar o cursor de cima da div, o background deve retornar à cor original. Utilize JavaScript para implementar essa funcionalidade e crie um efeito visual interessante ao alterar as cores.

DESAFIO PRÁTICO

Imagine que você está construindo um menu de navegação para um site. Cada item do menu é representado por um link em uma lista não ordenada (ul) com a classe 'menu-item'. Ao passar o mouse sobre cada item do menu, você deseja que o texto fique em negrito e a cor de fundo mude para destacar a seleção.

Além disso, quando o mouse sair do item do menu, tanto o negrito quanto a cor de fundo devem ser restaurados.

Utilize o evento 'mouseover' e 'mouseout' para implementar esse comportamento e melhore a usabilidade do menu de navegação.

SE LIGA NO CONTEÚDO DA PRÓXIMA AULA!

AULA 10 DE JAVASCRIPT.
PROJETO



INFINITY SCHOOL
VISUAL ART CREATIVE CENTER

A woman with long, dark hair and glasses is shown from the chest up, looking down at a laptop screen. The background is dark and slightly blurred.

IN

INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 09 - EVENTOS DOM II