Aula 3

Manipulação Avançada do DOM + Eventos Delegados

Prof. Bruno Torres

Objetivos da Aula

- Aprofundar o uso do DOM
- Criar e remover elementos dinamicamente
- Entender eventos delegados
- Praticar com mini-projeto de lista

Revisão: DOM

- Document Object Model
- Representa a estrutura HTML da página
- Usado para **manipular elementos** via JavaScript

Criando elementos

```
const novoItem = document.createElement("li");
novoItem.textContent = "Novo item";
```

Adicionando ao DOM

```
const lista = document.getElementById("lista");
lista.appendChild(novoItem);
```

- Adiciona elemento como último filho

Inserindo antes de outro

```
lista.insertBefore(novoItem, lista.firstChild);
```

- Insere antes de um filho específico

Removendo elementos

```
lista.removeChild(lista.firstChild);
```

- Remove um filho da lista

Exemplo prático

```
const btn = document.getElementById("adicionar");

btn.addEventListener("click", () => {
  const item = document.createElement("li");
  item.textContent = "Item novo";
  lista.appendChild(item);
});
```

Eventos em elementos criados

Problema comum:

```
elemento.addEventListener("click", () => {});
```

- Isso **não funciona** para elementos criados dinamicamente, **a menos que seja reatribuído**

Solução: Delegação de Eventos

- Usamos um **pai fixo**
- Verificamos o `event.target`

```
lista.addEventListener("click", (event) => {
  if (event.target.tagName === "LI") {
    event.target.remove();
  }
});
```

Vantagens da delegação

- Reduz número de eventListeners
- Funciona com elementos dinâmicos
- Mais performático

Estilizando elementos criados

```
item.classList.add("item-lista");
```

- Aplica classe para CSS ou lógica

Usando dataset

```
<li data-id="123">Tarefa</li>
```

```
let id = elemento.dataset.id;
```

- Acessa dados customizados no HTML

Mini projeto: Lista de tarefas

Funcionalidades:

- Adicionar tarefas
- Remover ao clicar
- Exibir total de itens

HTML base

```
<ul id="tarefas"></ul>
<input type="text" id="entrada" />
<button id="add">Adicionar</button>
```

JS funcional

```
add.addEventListener("click", () => {
  const li = document.createElement("li");
  li.textContent = entrada.value;
  tarefas.appendChild(li);
});
```

Melhorando: limpar input

```
entrada.value = "";
entrada.focus();
```

- Boa usabilidade

Remoção com clique (delegação)

```
tarefas.addEventListener("click", (e) => {
  if (e.target.tagName === "LI") {
    e.target.remove();
  }
});
```

Conclusão

- DOM permite manipular conteúdo dinamicamente
- Delegação de eventos é poderosa e eficiente
- Combinando tudo, criamos apps interativos

Obrigado!

Dúvidas?

Vamos praticar!