Removendo Tarefas Individuais em JavaScript

Por Escola Dnc

Introdução

Neste ebook, exploraremos como implementar a funcionalidade de remover tarefas individuais em uma aplicação de lista de tarefas usando JavaScript. Aprenderemos sobre a importância de IDs únicos, o uso do método filter em arrays, manipulação do DOM e boas práticas de programação para criar um código mais manutenível.

Criando a Função de Remoção

Definindo a Função RemoverTarefa

A base da nossa funcionalidade de remoção é a função `RemoverTarefa`. Vamos analisar sua estrutura e funcionamento:

```
function RemoverTarefa(id) {  tarefas = tarefas.filter(({ id: tarefaId }) => parseInt(tarefaId) !== parseInt(id));    const elementoLista = document.getElementById('toDoList');
elementoLista.removeChild(document.getElementById(`task-${id}`));}
```

Pontos importantes:

- A função recebe um `id` como parâmetro
- Utilizamos o método `filter` para criar um novo array sem a tarefa removida
- Convertemos os IDs para números com `parseInt` para evitar problemas de comparação
- Removemos o elemento do DOM usando `removeChild`

A Importância dos IDs Únicos

Os IDs desempenham um papel crucial nesta funcionalidade:

- Permitem identificar unicamente cada tarefa
- Facilitam a filtragem e remoção de tarefas específicas
- São essenciais para manipular elementos do DOM correspondentes às tarefas

Dica: Sempre use IDs únicos para elementos que precisarão ser manipulados individualmente.

Implementando o Botão de Remoção

Para que o usuário possa remover tarefas, precisamos adicionar um botão de remoção a cada item da lista. Vamos ver como fazer isso:

Criando o Botão

```
const removeTaskButton =
document.createElement('button');removeTaskButton.textContent =
'x';removeTaskButton.setAttribute('aria-label', 'Remover Tarefa');
```

Observações:

- Criamos um elemento `button` via JavaScript
- Definimos o texto do botão como "x"
- Adicionamos um `aria-label` para acessibilidade

Adicionando o Evento de Clique

```
removeTaskButton.onclick = () => RemoverTarefa(task.id);
```

- Utilizamos uma arrow function para chamar `RemoverTarefa` com o ID correto

Inserindo o Botão no Item da Lista

```
todoItem.appendChild(removeTaskButton);
```

- Adicionamos o botão como filho do elemento `li` da tarefa

Integrando a Funcionalidade na Aplicação

Modificando a Função de Criação de Tarefas

Para que nossa funcionalidade de remoção funcione tanto para tarefas existentes quanto para novas tarefas, precisamos modificar a função que cria os itens da lista:

```
function criarItemLista(task) {  const todoItem = document.createElement('li');  todoItem.id = `task-${task.id}`;  todoItem.textContent = task.texto;    const removeTaskButton = document.createElement('button');  removeTaskButton.textContent = 'x';  removeTaskButton.setAttribute('aria-label', 'Remover Tarefa');  removeTaskButton.onclick = () => RemoverTarefa(task.id);  todoItem.appendChild(removeTaskButton);    return todoItem;}
```

Benefícios desta abordagem:

- Código mais DRY (Don't Repeat Yourself)
- Facilita a manutenção, pois alterações precisam ser feitas em apenas um lugar
- Garante consistência entre tarefas existentes e novas

Boas Práticas e Considerações

Centralização de Lógica

Ao centralizar a lógica de criação e remoção de tarefas em funções específicas, obtemos vários benefícios:

- **Manutenibilidade:** Alterações podem ser feitas em um único lugar
- **Consistência:** Garante que todas as tarefas sejam tratadas da mesma forma
- **Reusabilidade:** As mesmas funções podem ser usadas em diferentes partes da aplicação

Acessibilidade

Não esqueça de considerar a acessibilidade ao implementar funcionalidades:

- Use `aria-label` para descrever a função de elementos não textuais
- Certifique-se de que todas as funcionalidades podem ser acessadas via teclado

Persistência de Dados

No exemplo dado, as tarefas são "chumbadas" (hard-coded) e reaparecem ao recarregar a página. Em uma aplicação real, você deveria:

- Implementar alguma forma de persistência de dados (localStorage, banco de dados, etc.)
- Atualizar o estado da aplicação ao remover tarefas para refletir as mudanças permanentemente

Conclusão

Neste ebook, exploramos como implementar a funcionalidade de remover tarefas individuais em uma aplicação JavaScript. Aprendemos sobre a importância de IDs únicos, manipulação de arrays com `filter`, criação dinâmica de elementos do DOM e boas práticas de programação.

Estes conceitos e técnicas formam uma base sólida para o desenvolvimento de aplicações web interativas e responsivas. À medida que você continua a desenvolver sua aplicação de lista de tarefas, considere adicionar mais funcionalidades, como marcar tarefas como concluídas ou implementar a remoção em massa de tarefas concluídas.

Lembre-se sempre de focar na manutenibilidade do código, na experiência do usuário e na acessibilidade ao implementar novas funcionalidades. Com estes princípios em mente, você estará bem preparado para enfrentar desafios mais complexos no desenvolvimento web.