Tarefas de DOM para To-Do List

Abaixo estão as tarefas que você deve completar para praticar a manipulação do DOM. Cada tarefa foi desenhada para ser progressivamente mais desafiadora e para cobrir uma variedade de funcionalidades úteis.

1. Adicionar uma nova tarefa

Descrição: Crie uma função JavaScript que adicione um item na lista de tarefas (`ul`) com o texto inserido no campo de input (`input`). A tarefa deve ser adicionada como um novo item (`li`) dentro da lista, e o campo de input deve ser limpo após a inserção da tarefa.

Objetivo: Aprender como manipular elementos de lista e usar `createElement` para adicionar novos elementos.

2. Remover uma tarefa

Descrição: Cada tarefa deve ter um botão "Remover" ao lado dela. Quando o botão for clicado, a tarefa correspondente deve ser removida da lista. Para isso, crie um botão dentro de cada item da lista e adicione um evento de clique que remova o item da lista quando acionado.

Objetivo: Praticar como adicionar eventos aos elementos e remover elementos do DOM.

3. Marcar tarefa como concluída

Descrição: Adicione um evento que permita ao usuário marcar uma tarefa como concluída. Ao clicar sobre uma tarefa, altere seu estilo, por exemplo, aplicando uma classe que muda a cor de fundo ou adiciona um efeito de texto tachado. A tarefa deve voltar ao estilo original caso seja clicada novamente.

Objetivo: Trabalhar com `classList.toggle` para adicionar e remover classes dinamicamente.

4. Limpar a lista de tarefas

Descrição: Crie uma função que apague todas as tarefas da lista de uma vez, utilizando um botão "Limpar Lista". O evento de clique nesse botão deve remover todos os itens da lista de tarefas de uma vez.

Objetivo: Aprender a manipular o conteúdo de um contêiner (no caso, a lista) e removê-lo de forma eficiente.

5. Contar o número de tarefas

Descrição: Após cada adição ou remoção de tarefa, mostre o número total de tarefas (pendentes e concluídas) na página. Atualize o contador sempre que uma tarefa for adicionada ou removida.

Objetivo: Manipular e atualizar o DOM com base em interações. A contagem de tarefas ajuda a consolidar o aprendizado sobre como interagir com elementos da página.

6. Alterar o texto de uma tarefa

Descrição: Crie uma função que permita ao usuário editar o texto de uma tarefa ao clicar sobre ela. Quando o usuário clicar em uma tarefa, um prompt ou campo de input deve aparecer para que ele possa digitar um novo texto. O texto da tarefa será alterado para o novo valor fornecido pelo usuário.

Objetivo: Trabalhar com edição de conteúdo dinâmico utilizando eventos de clique e input de texto.

7. Adicionar animação ao adicionar/remover tarefas

Descrição: Adicione uma animação suave (por exemplo, um fade-in ou slide) ao adicionar ou remover tarefas. Isso pode ser feito utilizando transições CSS. Quando uma tarefa for adicionada, ela deve aparecer suavemente, e quando for removida, deve desaparecer antes de ser retirada da lista.

Objetivo: Explorar animações CSS para tornar a interação do usuário mais dinâmica e agradável.

8. Adicionar uma tarefa com a tecla "Enter"

Descrição: Permita que o usuário adicione uma nova tarefa pressionando a tecla "Enter" no teclado, além de clicar no botão "Adicionar". A interação do teclado torna a interface mais acessível e conveniente.

Objetivo: Aprender a capturar eventos de teclado no JavaScript.

9. Tornar as tarefas editáveis diretamente na lista

Descrição: Em vez de usar um prompt, permita que o usuário edite o texto de uma tarefa diretamente na lista. Quando o usuário clicar em uma tarefa, o texto dela deve ser substituído por um campo de input para edição. Após a edição, o campo de input deve ser substituído pelo novo texto.

Objetivo: Aprender a substituir um elemento por outro dentro do DOM e como fazer alterações dinâmicas na página.

10. Estilizar tarefas concluídas de maneira distinta

Descrição: Aplique estilos específicos (como fundo colorido ou texto tachado) nas tarefas concluídas. O estilo deve ser alterado automaticamente quando a tarefa for marcada como concluída e revertido quando a tarefa for desmarcada.

Objetivo: Trabalhar com manipulação de classes CSS para modificar o estilo de um elemento de acordo com interações do usuário.

11. Adicionar um filtro de tarefas (pendentes/concluídas)

Descrição: Crie um filtro para exibir apenas as tarefas concluídas ou apenas as pendentes. O usuário deve poder selecionar qual filtro deseja ver (ex: "Mostrar Pendentes", "Mostrar Concluídas", "Mostrar Todas").

Objetivo: Trabalhar com eventos de filtro e lógica condicional para mostrar ou ocultar elementos baseados em classes ou atributos.

12. Adicionar uma tarefa com prioridade (Baixa, Média, Alta)

Descrição: Crie um sistema de prioridade para as tarefas. Adicione uma opção para o usuário escolher a prioridade de uma tarefa (Baixa, Média ou Alta) ao adicioná-la. A prioridade deve ser visível ao lado da tarefa na lista, e as tarefas de alta prioridade devem aparecer no topo da lista.

Objetivo: Aprender a adicionar atributos personalizados e como ordenar elementos com base em atributos.

13. Salvar tarefas no localStorage

Descrição: Salve as tarefas no `localStorage` para que elas não sejam perdidas ao recarregar a página. Quando a página for recarregada, as tarefas salvas no armazenamento local devem ser recuperadas e exibidas na lista.

Objetivo: Aprender como persistir dados no navegador usando `localStorage`.

14. Alterar a cor do botão "Adicionar" com base no preenchimento do campo

Descrição: Altere a aparência do botão "Adicionar" conforme o estado do campo de texto:

- **Campo vazio**: O botão "Adicionar" deve ficar desativado, com uma cor cinza e o cursor deve ser alterado para "não permitido" (sem interação).
- **Campo preenchido**: O botão deve ficar ativo, com uma cor vibrante (verde) e o cursor alterado para "pointer", permitindo interação.

Objetivo: Aprender como modificar a aparência de elementos interativos com base no estado de outros elementos na página.

15. Alterar cores de fundo por nível de prioridade (Baixa, Média, Alta)

Descrição: Modifique o fundo de cada tarefa com base na prioridade escolhida pelo usuário:

- **Baixa prioridade**: cor de fundo verde claro.
- **Média prioridade**: cor de fundo amarela.
- **Alta prioridade**: cor de fundo vermelha clara.

Objetivo: Praticar o uso de classes dinâmicas e estilos condicionais com base na interação do usuário.

16. Alterar o fundo da página dinamicamente

Descrição: Altere a cor de fundo do contêiner principal com base no número de tarefas na lista:

- **Lista vazia**: O fundo da página deve ter um tom mais claro, indicando que não há tarefas.
- **Lista com tarefas**: O fundo deve voltar ao tom padrão (cinza), indicando que há tarefas na lista.

Objetivo: Aprender como manipular o estilo do contêiner da página em resposta a alterações no conteúdo da lista.

17. Alterar a cor do botão "Remover" com base no status da tarefa

Descrição: A cor do botão "Remover" deve ser alterada conforme o status da tarefa (pendente ou concluída):

- **Tarefa pendente**: O botão de remover deve ter a cor vermelha padrão.
- **Tarefa concluída**: O botão deve mudar para verde, sinalizando que a tarefa foi finalizada.

Objetivo: Aprender a alterar o estilo de um botão com base na interação do usuário e no estado de um item.

18. Estilizar tarefas concluídas

Descrição: Tarefas concluídas devem ter uma aparência distinta:

- **Fundo mais suave**: Ao concluir uma tarefa, ela deve receber um fundo mais claro (suave), indicando que foi finalizada.
- **Texto riscado**: O texto da tarefa deve ser riscado, dando um indicativo visual de que foi concluída.

Objetivo: Praticar a alteração de estilos dinamicamente usando classes CSS para marcar tarefas como concluídas.

19. Alterar a cor do texto de acordo com a prioridade

Descrição: Aplique cores de texto diferentes com base na prioridade da tarefa:

- **Alta prioridade**: texto em vermelho.
- **Média prioridade**: texto em laranja.
- **Baixa prioridade**: texto em verde.

Objetivo: Aprender a modificar a cor do texto de elementos com base em atributos ou estados dinâmicos.

20. Adicionar transições suaves ao adicionar e remover tarefas

Descrição: Crie animações suaves para melhorar a interação do usuário:

- **Fade-in ao adicionar uma tarefa**: Quando uma nova tarefa for adicionada, ela deve aparecer suavemente.
- **Fade-out ao remover uma tarefa**: Quando uma tarefa for removida, ela deve desaparecer suavemente antes de ser removida do DOM.

Objetivo: Aprender como usar transições CSS para criar animações visuais que tornam a interação mais fluida.