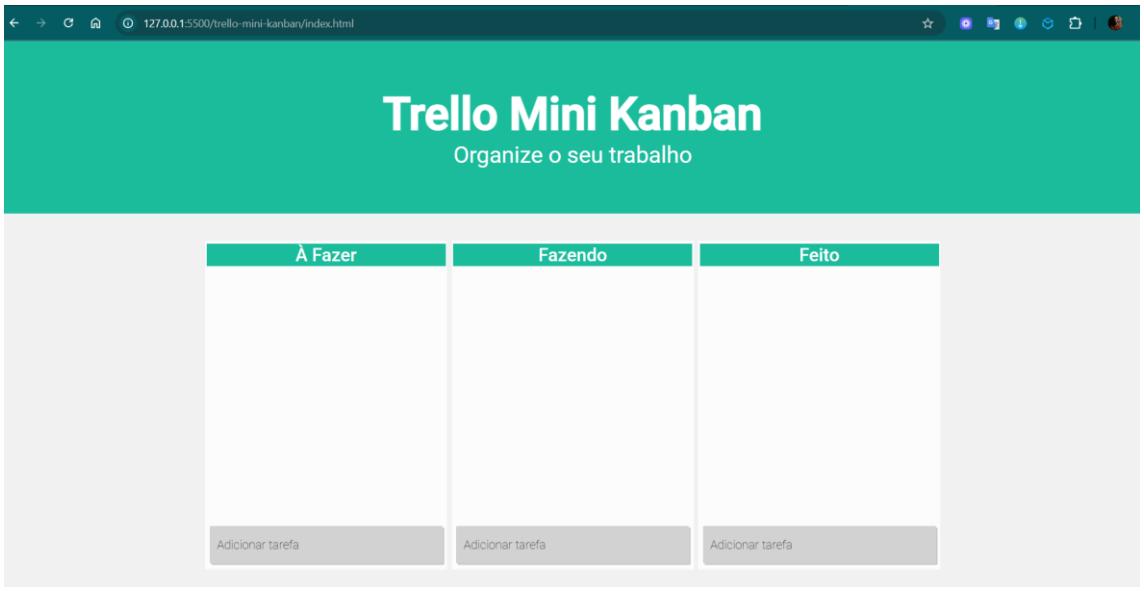

Roteiro de Ensino: Construindo um Mini Kanban com HTML, CSS e JavaScript

Objetivo Geral do Projeto

Construir um quadro de tarefas Kanban simples e funcional, que permita criar, remover e mover tarefas entre as colunas "À Fazer", "Fazendo" e "Feito". O projeto aborda a integração das três tecnologias essenciais da web.

Etapa 1: Estrutura da Página com HTML

- **Conceito da Aula:** Entender como o HTML define a estrutura e o conteúdo de uma página web.
- **Tempo Sugerido:** 50 minutos.
- **Passo a Passo (Live Coding):**
    1. **Estrutura Base:** Comece com a estrutura básica de um documento HTML (<!DOCTYPE html>, <html>, <head>, <body>). Explique o papel de cada tag.
    2. **Meta Tags e Links:** Insira as meta tags de charset e viewport. Explique a importância do viewport para a responsividade. Adicione o link para o CSS (style.css) e o link externo para a fonte e o Font Awesome (mesmo que não seja usado, é um bom exemplo de como incluir recursos externos).
    3. **Header:** Crie a seção <header> com um <h1> e um <p>. Mostre como o <h1> é um título de alta hierarquia.
    4. **Seção Principal (<main>):** Crie a tag <main> para conter todo o conteúdo principal do quadro Kanban.
    5. **Divisões do Quadro ( ):** Dentro da <main>, crie as três divs para as colunas ("À Fazer", "Fazendo", "Feito"). Explique o conceito de div como um contêiner genérico e o uso da classe .board para estilização.
    6. **Títulos e Listas:** Dentro de cada .board, adicione um <h3> para o título da coluna e uma tag <ul> (lista não ordenada) com o ID correspondente

(id="a-fazer", id="fazendo", id="feito"). Explique que o ul é onde as tarefas serão listadas.

7. **Botão "Adicionar Tarefa":** Dentro de cada .board, adicione um com um <p> para o texto. Explique que esta será nossa "área clicável" para adicionar tarefas.
8. **Tags Especiais de Drag-and-Drop:** Adicione os atributos ondrop, ondragover nas tags <ul> e onclick na div de "Adicionar tarefa". Explique que estes são "gatilhos" para o JavaScript, mas que a lógica virá depois.

- **Mini-Desafio da Aula:** Peça aos alunos para adicionarem uma quarta coluna, como "Em Revisão", usando as mesmas tags e classes.
- **Dever de Casa:** Peça para eles explorarem outras tags HTML semânticas (como <footer>, <article>).

Etapa 1: Estrutura da Página com HTML

- **Mini-Desafio da Aula:** Adicionar uma quarta coluna, como "Em Revisão", usando as mesmas tags e classes.
- **Resolução:**

```
<div class="board">
 <h3 class="title">Em Revisão</h3>
 <ul
  id="em-revisao"
  class="todo-list"
  ondrop="drop(event, this.id)"
  ondragover="over(event)"
 >
 </ul>
 <div class="add-card" onclick="addCard(this)">
  <p>Adicionar tarefa</p>
 </div>
</div>
```

Explicação:

- A solução envolve replicar a mesma estrutura HTML de uma coluna existente (...).
- É crucial manter a classe board para que a coluna herde os estilos CSS de layout e aparência.

- A tag <h3> é alterada para o novo título ("Em Revisão").
- O id da tag <ul> é alterado para um identificador único, como "em-revisao", que será usado pelo JavaScript para referenciar esta coluna.
- Os atributos de evento ondrop, ondragover e onclick devem ser mantidos para que a nova coluna seja funcional.

 Dever de Casa: Explorar outras tags HTML semânticas (como <footer>, <article>).

 Resolução e Explicação (Não há um código específico, mas uma explicação conceitual):

- **<footer>:** É usado para o rodapé da página ou de uma seção. Nele, geralmente colocamos informações como direitos autorais, links de contato ou navegação secundária. **Exemplo:** Adicionar <footer> para colocar o nome do autor do projeto ou uma licença.
- **<article>:** Representa um conteúdo independente e auto-contido, como um post de blog ou, no nosso caso, **cada cartão de tarefa poderia ser semanticamente um <article>**. Isso daria um significado mais específico ao conteúdo de cada item. **Exemplo:**

```
<li id="..." draggable="true" ondragstart="drag(event)">
  <article class="card-content">
    <p>Minha tarefa</p>
    <p class="remove" onclick="removeCard(this)">x</p>
  </article>
</li>
```

 seção e aside: O main já é uma seção semântica, mas poderíamos, por exemplo, usar <aside> para um menu lateral, se o projeto fosse maior.

 Conclusão para o aluno: O uso de tags semânticas não muda a aparência do site sem CSS, mas dá significado ao conteúdo para os navegadores, mecanismos de busca e tecnologias de acessibilidade.

Etapa 2: Estilizando com CSS

- **Conceito da Aula:** Usar o CSS para dar vida e estilo à estrutura HTML. Entender seletores de classe, propriedades de layout e interatividade visual.
- **Tempo Sugerido:** 50 minutos.
- **Pré-requisito:** Os alunos devem ter o arquivo style.css vazio, pronto para ser preenchido.
- **Passo a Passo (Live Coding):**

1. **Setup Inicial:** Adicione a importação da fonte "Roboto" e os estilos universais (* box-sizing, margin).
2. **Estilo do Corpo e Cabeçalho:** Estilize o <body> (cor de fundo, fonte) e o <header> (padding, cor de fundo, alinhamento de texto).
3. **Layout Principal (<main>):** Use display: flex na tag <main> para que as colunas (.board) se organizem lado a lado. Explique como o flex-wrap e o justify-content funcionam.
4. **Estilizando as Colunas (.board):** Aplique os estilos de largura (width), cor de fundo, padding, gap e flex-direction: column.
5. **Títulos, Listas e Itens:** Estilize os títulos (.board .title), as listas (.board ul) e os itens de tarefa (.board ul li). Explique a herança e o seletor de descendência (ul li).
6. **Interatividade Visual:** Aplique a pseudo-classe :hover para os itens de tarefa (li) e o botão de adicionar (add-card). Mostre como a mudança de cor, sombra e cursor (cursor: move) melhora a experiência do usuário.

- **Mini-Desafio da Aula:** Peça aos alunos para mudarem a cor de fundo do cabeçalho e as cores dos títulos das colunas.
- **Dever de Casa:** Peça para eles adicionarem um ícone de lixeira em cada item de tarefa (<li>) usando o Font Awesome, que já está linkado no HTML, e estilizar. Isso os preparará para a função de remover do JavaScript.

Etapa 2: Estilizando com CSS

- **Mini-Desafio da Aula:** Mudar a cor de fundo do cabeçalho e as cores dos títulos das colunas.
- **Resolução:**

```
/* Alterar esta regra CSS existente para a nova cor */
header {
  padding: 60px;
  text-align: center;
  background: #3498db; /* Cor de fundo alterada para um azul, por exemplo */
  color: white;
  font-size: 30px;
}

/* Alterar esta regra CSS existente para a nova cor */
```

```
.board .title {
  background-color: #3498db; /* Cor de fundo alterada para azul, para combinar */
  color: white;
  text-align: center;
  font-size: 24px;
  font-weight: 500;
}
```

- **Explicação:**
    - A solução envolve a identificação das regras CSS corretas que controlam o cabeçalho (header) e os títulos das colunas (.board .title).
    - O aluno deve simplesmente alterar a propriedade background ou background-color nessas regras para um novo valor. Isso demonstra o poder dos seletores de classe e tag para controlar o estilo de elementos específicos de forma centralizada.

- **Dever de Casa:** Adicionar um ícone de lixeira em cada item de tarefa (<li>) usando o Font Awesome e estilizar.
- **Resolução (HTML e CSS):**
    - **HTML (alterar no script.js):**

```
// Código JavaScript da função addCard()
const template = `
  <li id="${new Date().getTime()}" draggable="true" ondragstart="drag(event)">
    <p>${text}</p>
    <i class="fa fa-trash-o remove" onclick="removeCard(this)"></i>
  </li>
`;
```

- CSS (adicionar ou alterar regra):

```
/* Adicionar esta nova regra CSS para estilizar o ícone */
.board .todo-list .remove.fa {
  color: #c3c3c3;
  font-size: 14px;
  margin-left: auto; /* Alinha o ícone à direita */
```

```
  display: flex;
  align-items: center;
  justify-content: center;
}

/* Mudar a cor do hover para vermelho */
.board .todo-list .remove.fa:hover {
  color: red; /* Cor do ícone muda para vermelho ao passar o mouse */
}
```

Explicação:

- **HTML:** A solução não está diretamente no arquivo HTML, mas na **função addCard do JavaScript**, que é onde os itens de tarefa são criados dinamicamente. O aluno deve substituir o x por um elemento <i> do Font Awesome com a classe fa fa-trash-o.
- **CSS:** É necessário adicionar uma nova regra para estilizar o ícone. O seletor .remove.fa garante que a regra se aplique apenas ao ícone de lixeira. A propriedade margin-left: auto é uma forma elegante de empurrar o ícone para a direita, alinhado com o Flexbox pai. O hover é ajustado para dar uma cor mais forte.

Etapa 3: Lógica e Interatividade com JavaScript

- **Conceito da Aula:** Entender como o JavaScript dá vida à página, manipulando o HTML e respondendo a eventos do usuário. Foco em Funções, DOM e eventos de drag-and-drop.
- **Tempo Sugerido:** 50 minutos.
- **Pré-requisito:** Os alunos devem ter um arquivo script.js vazio, linkado no HTML.
- **Passo a Passo (Live Coding):**
    1. **Função de Adicionar Tarefa (addCard):**
        - Comece explicando a função addCard.
        - Explique o prompt() para coletar o texto da tarefa.
        - Mostre como obter a ul correta (o board) usando element.previousElementSibling.id e document.getElementById().
        - **A Magia do Template String:** Apresente o conceito de template string (` `) e como ele permite criar um bloco HTML dinâmico,

inserindo variáveis (${text}, ${new Date().getTime()}). Explique por que o id dinâmico é importante.

- **Inserção no HTML:** Mostre como usar board.innerHTML += template para adicionar o novo item à lista.

2. **Função de Remover Tarefa (removeCard):**
    - Explique a função removeCard(element).
    - Mostre como o element.parentElement pega o li pai do botão x.
    - Demonstre a função .remove() para apagar o elemento da página.
3. **Lógica de Arrastar e Soltar (Drag-and-Drop):**
    - **drag(event):** Explique o evento ondragstart no HTML. Na função JS, mostre como usar event.dataTransfer.setData() para "guardar" o ID da tarefa que está sendo arrastada.
    - **over(event):** Explique que o evento ondragover é necessário para permitir que um elemento seja solto. A única linha de código, event.preventDefault(), é o ponto principal.
    - **drop(event, id):**
        - Explique o evento ondrop. Mostre como obter o ID do card arrastado com event.dataTransfer.getData().
        - Mostre como obter a ul de destino com document.getElementById(id).
        - A linha target.appendChild(card) é a mágica que move o elemento. Explique que o appendChild apenas move o elemento de sua posição original para a nova.

- **Mini-Desafio da Aula:** Peça para eles alterarem a função addCard para que a tarefa seja adicionada apenas se o usuário digitar algo (evitar tarefas em branco). Use um if (text) {...}.
- **Dever de Casa:** Peça para eles pesquisarem sobre o localStorage e tentarem salvar as tarefas no navegador para que não se percam quando a página for recarregada.

Etapa 3: Lógica e Interatividade com JavaScript

- **Mini-Desafio da Aula:** Alterar a função addCard para que a tarefa seja adicionada apenas se o usuário digitar algo (evitar tarefas em branco).
- **Resolução:**

```
function addCard(element) {
```

```
  const ulId = element.previousElementSibling.id;
  const text = prompt("Qual é a tarefa?");

  // Adicionar esta verificação
  if (text && text.trim() !== "") {
   const board = document.getElementById(ulId);

   const template = `
    <li id="${new Date().getTime()}" draggable="true" ondragstart="drag(event)">
     <p>${text}</p>
     <p class="remove" onclick="removeCard(this)">x</p>
    </li>
   `;
   board.innerHTML += template;
  }
}
```

Explicação:

- A solução envolve um simples condicional if em torno da lógica de criação do cartão.
- A condição if (text && text.trim() !== "") verifica duas coisas:
    1. text: Se a variável text existe (ou seja, se o usuário não cancelou o prompt, que retornaria null).
    2. text.trim() !== "": Se o texto, após remover os espaços em branco do início e do fim (.trim()), não é uma string vazia.
- Se a condição for verdadeira, o código de criação do cartão é executado. Caso contrário, nada acontece, e nenhuma tarefa em branco é criada.

 Dever de Casa: Pesquisar sobre o localStorage e tentar salvar as tarefas no navegador para que não se percam quando a página for recarregada.

 Resolução (Conceitual, com indicação de código):

- **Conceito:** O localStorage é uma API do navegador que permite armazenar dados-chave-valor (apenas strings) de forma persistente, mesmo após o navegador ser fechado.
- **Etapas para o Aluno (Orientação):**

1. **Salvando Dados:** Na função addCard e removeCard, o aluno precisaria atualizar um array de objetos JavaScript que representa o estado do quadro Kanban. Ao final de cada alteração (adicionar, remover, mover), esse array seria salvo no localStorage usando localStorage.setItem('kanbanData', JSON.stringify(meuArray)). É necessário usar JSON.stringify porque o localStorage só armazena strings.
2. **Carregando Dados:** Ao carregar a página (idealmente, com uma nova função loadKanban()), o aluno precisaria verificar se existem dados no localStorage com a chave 'kanbanData'.
3. **Montando o HTML:** Se os dados existirem, eles seriam lidos com localStorage.getItem('kanbanData') e convertidos de volta para um array de objetos usando JSON.parse(). Em seguida, o código percorreria este array e criaria dinamicamente os elementos <li> para cada tarefa, adicionando-os às colunas corretas.

Exemplo de Código (para guia):

```
// Código de exemplo para a função de carregar
function loadKanban() {
  const savedData = localStorage.getItem('kanbanData');
  if (savedData) {
    const data = JSON.parse(savedData);
    // ... Lógica para iterar sobre o array 'data' e criar os elementos HTML
    // para cada tarefa, adicionando-os às suas respectivas <ul>.
  }
}

// Ao carregar a página
window.onload = loadKanban;

// Código de exemplo para a função de salvar (chamada após adicionar, remover, etc.)
function saveKanban() {
  // Obter todas as tarefas da página
  // Salvar como JSON no localStorage
  // Ex: localStorage.setItem('kanbanData', JSON.stringify(tarefas));
}
```

Conclusão: Este dever de casa introduz um conceito fundamental de desenvolvimento web: persistência de dados no lado do cliente. É um desafio mais complexo, mas extremamente valioso.

Com essas resoluções e explicações detalhadas, você tem um material completo para guiar seus alunos de forma eficiente e segura em cada etapa do projeto.

Este roteiro é sequencial e cada aula constrói sobre a anterior, culminando em um projeto totalmente funcional no final. A sua abordagem de "mini-projetos" é a base para o sucesso. O código que você forneceu é perfeito para isso!

@douglasabnovato

01/04/2025