Atividade: Construindo uma Página Interativa com JavaScript

Hoje vamos criar página web do zero e adicionar, passo a passo, diferentes funcionalidades interativas usando JavaScript. Vamos começar com uma página em branco e terminar com o "Painel de Testes JavaScript" completo.

Preparação: A Estrutura Base

Antes de começar com o JavaScript, precisamos de uma estrutura HTML e um pouco de CSS.

1. **Crie um ficheiro** chamado index.html.
2. **Copie e cole o código abaixo** para ter a nossa base. Ele já inclui o layout e os estilos com Tailwind CSS, para que nos possamos focar apenas no JavaScript.

```
<!DOCTYPE html>
<html lang="pt-PT">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Testes de Funcionalidades JavaScript</title>
  <link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;700&display=swap"
rel="stylesheet">
  <script src="https://cdn.tailwindcss.com"></script>
  <style>
    body { font-family: 'Inter', sans-serif; }
    button { transition: transform 0.1s ease-in-out; }
    button:hover { transform: translateY(-2px); }
  </style>
</head>
<body class="bg-gray-50 text-gray-800">
  <div class="container mx-auto p-8 max-w-4xl">
    <header class="text-center mb-10">
      <h1 class="text-4xl font-bold text-gray-900">Painel de Testes JavaScript</h1>
      <p class="text-lg text-gray-600 mt-2">Uma coleção de testes interativos para explorar
funcionalidades do JS.</p>
    </header>
    <main class="grid grid-cols-1 md:grid-cols-2 gap-8">
      <!-- Os nossos testes serão adicionados aqui -->
    </main>
  </div>
  <script>
    // O nosso código JavaScript virá aqui!
  </script>
</body>
</html>
```

Passo 1: Manipular o DOM - O "Olá, Mundo!" Interativo

O primeiro passo é fazer o JavaScript interagir com o HTML. Vamos adicionar um botão que altera um texto na página.

1. **Adicione o HTML** do primeiro teste dentro da tag <main>:

```
<!-- Teste 1: Manipulação do DOM -->
<div class="bg-white p-6 rounded-xl shadow-sm border border-gray-200">
    <h2 class="text-2xl font-semibold mb-4 border-b pb-2">1. Manipulação do DOM</h2>
    <p id="text-to-change" class="mb-4 p-3 bg-gray-100 rounded-md">Este texto vai mudar.</p>
    <div class="flex flex-wrap gap-2">
        <button id="change-text-btn" class="bg-blue-500 text-white px-4 py-2 rounded-lg
font-medium">Alterar Texto</button>
    </div>
</div>
```

2. **Adicione o JavaScript** correspondente dentro da tag <script>:

```
document.addEventListener('DOMContentLoaded', () => {
    // Seleciona o parágrafo e o botão pelos seus IDs
    const textToChange = document.getElementById('text-to-change');
    const changeTextBtn = document.getElementById('change-text-btn');

    // Adiciona um "ouvinte" que espera por um clique no botão
    changeTextBtn.addEventListener('click', () => {
        // Quando o botão for clicado, altera o conteúdo do texto
        textToChange.textContent = 'Texto alterado com sucesso pelo JavaScript!';
    });
});
```

Teste: Abra o ficheiro index.html no navegador e clique no botão "Alterar Texto". O texto deve mudar!

Passo 2: Alterar Estilos Dinamicamente

Agora, vamos adicionar outro botão que manipula o CSS do mesmo texto.

1. **Adicione um novo botão** ao HTML do Teste 1, ao lado do botão existente:

```
<button id="change-style-btn" class="bg-teal-500 text-white px-4 py-2 rounded-lg
font-medium">Alterar Estilo</button>
```

2. **Adicione o JavaScript** para este novo botão dentro do DOMContentLoaded:

```
// (Adicione este código abaixo do código do passo anterior)
document.getElementById('change-style-btn').addEventListener('click', () => {
    textToChange.classList.toggle('bg-teal-100');
```

```
    textToChange.classList.toggle('text-teal-800');
    textToChange.classList.toggle('font-bold');
});
```

Teste: Clique no botão "Alterar Estilo" várias vezes. Deverá ver o estilo do texto a ser adicionado e removido.

Passo 3: Reagir a Eventos do Rato e Teclado

Vamos criar um novo cartão de teste para explorar outros tipos de eventos do utilizador.

1. **Adicione o HTML** para o Teste 2 dentro da tag <main>:

```
<!-- Teste 2: Eventos do Utilizador -->
<div class="bg-white p-6 rounded-xl shadow-sm border border-gray-200">
    <h2 class="text-2xl font-semibold mb-4 border-b pb-2">2. Eventos do Utilizador</h2>
    <div id="mouse-box" class="h-24 bg-gray-100 rounded-md flex items-center justify-center
cursor-pointer mb-4 select-none">Passe o rato aqui</div>
    <p class="text-sm text-gray-600">Pressione uma tecla para ver o seu código:</p>
    <p class="text-lg font-mono p-2 bg-gray-800 text-white rounded-md mt-1 h-10 flex items-center"
id="key-pressed-output"> </p>
</div>
```

2. **Adicione o JavaScript** para os eventos de rato e teclado:

```
// (Adicione este código dentro do DOMContentLoaded)
const mouseBox = document.getElementById('mouse-box');
mouseBox.addEventListener('mouseover', () => {
    mouseBox.textContent = 'O rato entrou!';
    mouseBox.classList.add('bg-blue-500', 'text-white');
});
mouseBox.addEventListener('mouseout', () => {
    mouseBox.textContent = 'Passe o rato aqui';
    mouseBox.classList.remove('bg-blue-500', 'text-white');
});

document.addEventListener('keydown', (event) => {
    document.getElementById('key-pressed-output').textContent = event.code;
});
```

Teste: Mova o rato para dentro e para fora da caixa cinzenta. Depois, clique na página e pressione algumas teclas.

Passo 4: Buscar Dados de uma API (JavaScript Assíncrono)

Este é um passo mais avançado. Vamos buscar dados de um servidor externo e exibi-los na nossa página.

1. **Adicione o HTML** para o Teste 3:

```
<!-- Teste 3: JavaScript Assíncrono -->
<div class="bg-white p-6 rounded-xl shadow-sm border border-gray-200 md:col-span-2">
  <h2 class="text-2xl font-semibold mb-4 border-b pb-2">3. JavaScript Assíncrono (Fetch API)</h2>
  <p class="mb-4">Clique para buscar uma lista de utilizadores de uma API pública.</p>
  <button id="fetch-data-btn" class="bg-indigo-500 text-white px-4 py-2 rounded-lg
font-medium">Buscar Dados</button>
  <div id="api-data" class="mt-4 p-4 bg-gray-50 border rounded-md min-h-[100px]">
    Os dados da API aparecerão aqui...
  </div>
</div>
```

2. **Adicione o JavaScript** com a função async/await para buscar os dados:

```
// (Adicione este código dentro do DOMContentLoaded)
const fetchDataBtn = document.getElementById('fetch-data-btn');
const apiDataDiv = document.getElementById('api-data');

fetchDataBtn.addEventListener('click', async () => {
  apiDataDiv.innerHTML = '<p>A carregar dados...</p>';
  try {
    const response = await fetch('https://jsonplaceholder.typicode.com/users?_limit=5');
    if (!response.ok) throw new Error(`Erro na rede: ${response.status}`);
    const users = await response.json();
    const userList = users.map(user => `
      <div class="p-2 border-b last:border-b-0">
        <p class="font-semibold">${user.name}</p>
        <p class="text-sm text-gray-500">${user.email}</p>
      </div>
    `).join('');
    apiDataDiv.innerHTML = userList;
  } catch (error) {
    apiDataDiv.innerHTML = `<p class="text-red-500">Falha ao buscar dados: ${error.message}</p>`;
  }
});
```

Teste: Clique no botão "Buscar Dados" e aguarde que a lista de utilizadores apareça.