RESUMO DA AULA

16 Requisições com o Fetch

O fetch um m todo JavaScript moderno que permite realizar requisi es HTTP para APIs de maneira simplificada e pr tica, e que retorna uma Promise como resultado. Ele muito utilizado para buscar ou enviar informa es para APIs, e nos permite manipular as respostas de forma ass ncrona. Como funciona o fetch O m todo fetch uma funcionalidade nativa do JavaScript que recebe como argumento uma URL de um servi o web e, em resposta, nos entrega uma Promise. Essa Promise cont m o objeto de resposta da API, que pode ser transformado em um formato utiliz vel como JSON ou texto, utilizando m todos como .json() ou .text(). Aqui vamos fazer um exemplo pr tico de como utilizar o fetch para interagir com uma API p blica de cartas, chamada Deck of Cards API. Essa API permite, por exemplo, embaralhar um baralho de cartas e tirar cartas aleat rias para simular jogos como blackjack. 1. Criando um baralho embaralhado Primeiro, vamos criar uma fun o chamada criarBaralhoEmbaralhado que ser respons vel por criar e embaralhar um baralho utilizando o endpoint /deck/new/shuffle/ da API. Nessa fun o, vamos usar o fetch para realizar a requisi o GET para esse endpoint. async function criarBaralhoEmbaralhado() { const url = "https://deckofcardsapi.com/api/deck/new/shuffle/?deck_count=1"; const resposta = await fetch(url); return await resposta.json(); } Explica o: Usamos o await antes do fetch para esperar que a Promise seja resolvida e obter a resposta completa da API. Em seguida, convertemos a resposta para o formato JSON com resposta.json() para que possamos acessar as propriedades do objeto retornado. 2. Tirando uma carta aleat ria do baralho Depois de embaralhar o baralho, precisamos tirar uma carta aleat ria dele. Para isso, criamos a fun o tirarUmaCarta que utilizar o endpoint /deck/{deck_id}/draw/?count=1 da API. Nessa fun o, vamos passar o deck_id como par metro na URL. async function tirarUmaCarta(deckId) { const url = `https://deckofcardsapi.com/api/deck/${deckId}/draw/?count=1`; const resposta = await fetch(url); return await resposta.json(); } 3. Juntando tudo: Fun o principal para manipular as cartas Vamos

criar a fun o tirarCartaAleatoriaDoBaralho para orquestrar as chamadas e puxar uma carta aleat ria do baralho criado anteriormente. Essa fun o vai chamar a criarBaralhoEmbaralhado para obter o deck_id e, em seguida, utilizar a tirarUmaCarta para buscar a carta. async function tirarCartaAleatoriaDoBaralho() { const baralho = await criarBaralhoEmbaralhado(); const carta = await tirarUmaCarta(baralho.deck_id); console.log(carta); } 4. Exibindo a carta na tela Para tornar o exemplo mais interessante, vamos exibir a carta tirada diretamente no navegador. Para isso, vamos usar o valor retornado pela API, que inclui a URL de uma imagem da carta. No HTML, vamos criar uma tag <img> para mostrar a carta. HTML: <button id="puxar-carta">Puxar Carta</button> Agora, vamos adicionar um EventListener ao bot o para que, ao clicar, a carta seja puxada e exibida. JavaScript: document.getElementById('puxar-carta').addEventListener('click', async () => { const baralho = await criarBaralhoEmbaralhado(); const carta = await tirarUmaCarta(baralho.deck_id); const imagemCarta = carta.cards[0].image; // Pega a URL da imagem da carta document.getElementById('carta').src = imagemCarta; // Atualiza a imagem no HTML }); 5. Explicando o fluxo completo criarBaralhoEmbaralhado: Faz uma requisi o para a API criar um novo baralho embaralhado e retorna o deck_id. tirarUmaCarta: Utiliza o deck_id retornado e chama o endpoint para retirar uma carta espec fica. tirarCartaAleatoriaDoBaralho: Orquestra o fluxo, chamando as duas fun es anteriores e imprimindo a carta no console. EventListener no bot o: Quando o bot o clicado, busca uma nova carta aleat ria e atualiza a imagem na tela. Conclus o O fetch uma maneira poderosa e simples de fazer requisi es HTTP no JavaScript. Com ele, conseguimos interagir com APIs para buscar ou enviar informa es de forma pr tica e intuitiva. Usar fetch com async e await torna o c digo mais leg vel e f cil de gerenciar, especialmente em projetos que utilizam muitas requisi es ass ncronas.
