RESUMO DA AULA

13 Tratando erros no JavaScript

Tratar erros no JavaScript uma habilidade essencial para garantir que a aplica o funcione corretamente mesmo quando ocorrem problemas inesperados. Nesta aula, vamos entender como tratar esses erros usando as instru es try, catch, finally e throw. Estrutura b sica do tratamento de erros Quando queremos monitorar e capturar erros em um bloco de c digo, utilizamos a estrutura try...catch. Veja a sintaxe b sica: try { // Bloco de c digo a ser executado } catch (err) { // Bloco de c digo que ser executado caso ocorra um erro } O bloco try cont m o c digo que queremos executar. Se ocorrer um erro em algum ponto dentro do try, o fluxo de execu o imediatamente transferido para o catch. A vari vel err (ou qualquer nome que escolhermos) captura o erro, e podemos acessar informa es sobre ele. Exemplo de try...catch: Vamos usar o c digo que fizemos na aula passada como exemplo: let ferverAgua = (chaleiraEstaNoFogao, fogaoEstaLigado) => { return new Promise((resolve, reject) => { if (chaleiraEstaNoFogao && fogaoEstaLigado) { console.log('Passo 1 finalizado: gua foi fervida'); resolve(); } else { let mensagemDeErro = 'Erro: necess rio colocar a chaleira com a gua e ligar o fog o sen o teu caf zinho n o vai ficar pronto.'; reject(mensagemDeErro); } }); }; let chaleiraEstaNoFogao = true; let fogaoEstaLigado = false; async function iniciarProcessoDeFazerCafe() { try { const aguaFervida = await ferverAgua(chaleiraEstaNoFogao, fogaoEstaLigado); console.log(' gua est pronta.'); } catch (err) { console.error(`Erro ao tentar ferver a gua: ${err}`); } } iniciarProcessoDeFazerCafe(); Nesse exemplo, o try tenta executar a chamada para ferverAgua, mas como fogaoEstaLigado false, a Promise rejeitada e o fluxo imediatamente transferido para o catch. A vari vel err recebe a mensagem de erro que foi passada no reject. O bloco finally Al m do try e do catch, podemos usar o finally. Esse bloco opcional e ser executado sempre, independentemente de ter ocorrido um erro ou n o. til para liberar recursos, fechar conex es com banco de dados ou realizar qualquer opera o que precise acontecer ap s a execu o de um bloco

try...catch. async function iniciarProcessoDeFazerCafe() { try { const aguaFervida = await ferverAgua(chaleiraEstaNoFogao, fogaoEstaLigado); console.log(' gua est pronta.'); } catch (err) { console.error(`Erro ao tentar ferver a gua: ${err}`); } finally { console.log('Processo de preparo de caf finalizado.'); } } iniciarProcessoDeFazerCafe(); No exemplo acima, o bloco finally ser executado mesmo se o catch capturar um erro. Isso garante que mensagens de conclus o ou recursos sejam sempre tratados corretamente. Lan ando exce es com throw s vezes, queremos lan ar nossos pr prios erros para capturar situa es que o JavaScript n o considera um erro automaticamente. Para isso, usamos a instru o throw, que "arremessa" um erro para ser capturado pelo catch. Vamos adicionar um exemplo em que for amos um erro caso os par metros passados para ferverAgua n o sejam do tipo booleano: let ferverAgua = (chaleiraEstaNoFogao, fogaoEstaLigado) => { return new Promise((resolve, reject) => { if (typeof chaleiraEstaNoFogao !== "boolean" || typeof fogaoEstaLigado !== "boolean") { throw new Error("Par metros devem ser booleanos."); } if (chaleiraEstaNoFogao && fogaoEstaLigado) { console.log('Passo 1 finalizado: gua foi fervida'); resolve(); } else { let mensagemDeErro = 'Erro: necess rio colocar a chaleira com a gua e ligar o fog o sen o teu caf zinho n o vai ficar pronto.'; reject(mensagemDeErro); } }); }; let chaleiraEstaNoFogao = "true"; // Intencionalmente passamos um valor de string let fogaoEstaLigado = true; async function iniciarProcessoDeFazerCafe() { try { const aguaFervida = await ferverAgua(chaleiraEstaNoFogao, fogaoEstaLigado); console.log(' gua est pronta.'); } catch (err) { console.error(`Erro ao tentar ferver a gua: ${err}`); } finally { console.log('Processo de preparo de caf finalizado.'); } } iniciarProcessoDeFazerCafe(); Quando executamos o exemplo acima, o throw for a a interrup o do fluxo normal e passa o controle diretamente para o catch. Isso muito til para validar dados ou impedir que a aplica o continue com valores incorretos. Tipos de erros no JavaScript Al m de lan ar exce es personalizadas, o JavaScript possui uma s rie de construtores de erros que podemos utilizar: Error: erro gen rico. SyntaxError: erro de sintaxe (normalmente ocorre quando o c digo mal formatado). TypeError: erro de tipo (ocorre quando uma opera o realizada em um valor de tipo incorreto). ReferenceError: erro de refer ncia (ocorre

quando tentamos acessar uma vari vel que n o existe). RangeError: erro de intervalo (ocorre quando um valor n o est dentro do intervalo permitido). Conclus o Nesta aula, aprendemos a: Capturar e tratar erros usando try...catch. Executar um bloco de c digo com finally que ser executado independentemente de erros. Lan ar erros personalizados usando throw. Utilizar construtores de erros nativos do JavaScript para definir o tipo exato de erro. O tratamento de erros essencial para evitar que problemas pequenos causem falhas graves em uma aplica o. Com esses recursos, podemos capturar problemas e lidar com eles de maneira apropriada, garantindo uma experi ncia mais est vel e controlada para o usu rio. Na pr xima aula, vamos entender o que s o APIs e como utiliz -las para conectar nossa aplica o a servi os externos.
