RESUMO DA AULA

9 Síncrono vs Assíncrono

Nesta aula, vamos explorar os conceitos de c digo s ncrono e c digo ass ncrono no JavaScript. Esse um tema fundamental para quem deseja trabalhar com JavaScript de forma mais avan ada, pois ele define como as opera es s o executadas e como lidamos com fluxos de c digo que podem levar um tempo vari vel para serem conclu dos. O que c digo S ncrono? Um c digo considerado s ncrono quando as instru es s o executadas uma de cada vez, em uma sequ ncia linear. Ou seja, uma linha s executada ap s a conclus o da linha anterior. Esse tipo de execu o cria uma depend ncia l gica entre as opera es e garante que o fluxo do c digo siga um padr o previs vel. Exemplo de fluxo s ncrono: function adicionarA ucar(){ console.log("Adicionando a car..."); } function prepararCaf () { console.log("Preparando o caf ..."); } adicionarA ucar(); // Executa primeiro prepararCaf (); // S executa ap s a execu o de adicionarA ucar Neste exemplo, o JavaScript segue a execu o linha por linha. Primeiro, ele executa a fun o adicionarA ucar e, s depois, executa prepararCaf . O c digo espera a conclus o de uma tarefa para passar pr xima. O que c digo Ass ncrono? Em um c digo ass ncrono, as instru es podem ser executadas fora de ordem, e uma instru o n o depende da conclus o das anteriores para ser iniciada. Isso permite que opera es que podem levar mais tempo como chamadas de API, leitura de arquivos ou consultas a bancos de dados n o bloqueiem a execu o de outras partes do programa. Exemplo de fluxo ass ncrono: function adicionarA ucar() { console.log("Adicionando a car..."); } function prepararCaf () { setTimeout(() => { console.log("Preparando o caf ..."); // Executa ap s 3 segundos }, 3000); } adicionarA ucar(); prepararCaf (); Neste exemplo, o setTimeout uma fun o ass ncrona. Mesmo que a linha prepararCaf () seja chamada ap s adicionarA ucar(), o console.log dentro do setTimeout executado 3 segundos depois, independentemente do fluxo do c digo. Isso uma demonstra o de como o JavaScript lida com opera es ass ncronas. Exemplificando com a analogia do caf : Imagine que voc quer fazer um caf e tem duas formas de realizar isso: Forma S ncrona: Voc coloca a gua

para ferver e fica parado olhando a gua esquentar at que ela esteja pronta. S depois disso voc pega o p de caf , coloca no filtro e come a a passar o caf . Forma Ass ncrona: Voc coloca a gua para ferver e, enquanto espera, j vai preparando o p , pegando a x cara e ajustando o filtro. Quando a gua estiver pronta, voc j est com tudo pronto para passar o caf . A execu o ass ncrona mais eficiente, pois voc aproveita o tempo de espera para adiantar outras tarefas. No c digo JavaScript, aplicamos essa l gica com fun es como setTimeout, setInterval, promises e async/await. Vantagens do C digo Ass ncrono: Efici ncia de Tempo: Permite que o programa continue executando outras tarefas enquanto aguarda o t rmino de opera es que levam mais tempo, como chamadas de rede. Melhor Desempenho: Opera es ass ncronas s o fundamentais para evitar bloqueios de execu o em projetos de larga escala e aplica es que processam muitas requisi es. Experi ncia de Usu rio: Ajuda a criar interfaces mais flu das, onde componentes de UI continuam respondendo, mesmo que processos mais demorados estejam sendo realizados em segundo plano. Controlando a Assincronicidade no C digo: No JavaScript, temos v rias formas de lidar com opera es ass ncronas. Alguns exemplos s o: setTimeout e setInterval: Para controlar a execu o de tarefas com atrasos espec ficos, como vimos anteriormente. Callbacks: Fun es passadas como argumento para outras fun es, para serem executadas ap s o t rmino de uma opera o. Promises: Uma forma moderna de tratar assincronismo, que torna o c digo mais leg vel e modular. Async/Await: Permite que escrevamos c digo ass ncrono como se fosse s ncrono, tornando a leitura e a manuten o do c digo muito mais simples. Exemplo Pr tico: Caf com Assincronismo Vamos implementar o exemplo do caf usando setTimeout para simular a fervura da gua de forma ass ncrona: function ferverAgua() { console.log('Colocando gua para ferver...'); setTimeout(() => { console.log(' gua ferveu, pronta para passar o caf .'); passarOCafe(); }, 5000); } function prepararTudoParaOCafe() { console.log('Pegando o p de caf ...'); console.log('Pegando o filtro...'); console.log('Colocando o caf no filtro...'); } function passarOCafe() { console.log('Passando o caf ...'); } // Executando as fun es ferverAgua(); prepararTudoParaOCafe(); Neste exemplo, enquanto a gua ferve (5 segundos simulados), o c digo n o fica "preso" esperando a gua ferver. Ele segue executando a pr xima fun o (prepararTudoParaOCafe) e, quando a gua est

pronta, a fun o passarOCafe chamada. Resumo: C digo S ncrono: Executa opera es de forma linear, uma de cada vez, seguindo a ordem de execu o das instru es. C digo Ass ncrono: Permite que o programa execute outras opera es enquanto espera por resultados de processos demorados, como leitura de arquivos ou chamadas de rede. Na pr xima aula, vamos aprofundar esse conceito aprendendo sobre Promises, a forma mais moderna e pr tica de trabalhar com c digo ass ncrono em JavaScript.
