RESUMO DA AULA

8 SetInterval e SetTimeout

Nessa aula vamos aprender a usar duas funções muito importantes no JavaScript: setInterval e setTimeout. Essas funções são utilizadas para controlar a execução de blocos de código após um período de tempo específico, permitindo que você execute algo uma única vez ou em intervalos de tempo regulares.

Essas duas funções são higher-order functions, pois recebem outra função como parâmetro (um callback) e as executam de forma assíncrona. Vamos ver em detalhes como cada uma funciona e quais são as suas principais diferenças.

Diferença entre setTimeout e setInterval

setTimeout: Executa um bloco de código apenas uma vez após um determinado período de tempo.

setInterval: Executa um bloco de código repetidamente em intervalos fixos de tempo, até que ele seja explicitamente interrompido.

Exemplo com setTimeout

A função setTimeout é usada para aguardar um tempo específico antes de executar um bloco de código. A sintaxe é a seguinte:

setTimeout(function, delay);

function: É a função de callback que será executada após o tempo definido.

delay: É o tempo de espera em milissegundos (1 segundo = 1000 milissegundos).

Exemplo de Uso

setTimeout(function () {

```
    alert('Executado após 3 segundos!');
}, 3000);
```

Nesse exemplo, um alert será exibido no navegador após 3 segundos (3000 milissegundos).

Se você rodar esse código, verá que a mensagem só aparece uma única vez, após o tempo definido. Isso é útil quando queremos executar algo apenas uma vez depois de um atraso.

Exemplo com setInterval

A função setInterval é usada para repetir a execução de um bloco de código em intervalos regulares de tempo. A sintaxe é semelhante à do setTimeout:

```
setInterval(function, interval);
```

function: É a função de callback que será executada em cada intervalo.

interval: É o tempo (em milissegundos) entre cada execução.

Exemplo de Uso

```
setInterval(function () {
    console.log('Executando a cada 2 segundos');
}, 2000);
```

Nesse exemplo, a mensagem "Executando a cada 2 segundos" será exibida no console a cada 2 segundos. O setInterval continua executando o código até que ele seja explicitamente interrompido ou que a página seja recarregada.

Comportamento Assíncrono

É importante entender que ambas as funções são assíncronas, ou seja, mesmo que você tenha código depois do setTimeout ou setInterval, ele será executado imediatamente e não aguardará a

execução dessas funções.

Vamos ver isso na prática:

```
console.log('Código antes do setTimeout');
setTimeout(function () {
    console.log('Executado dentro do setTimeout');
}, 3000);
console.log('Código depois do setTimeout');
```

Se você rodar esse exemplo, verá a seguinte sequência no console:

Código antes do setTimeout

Código depois do setTimeout

Executado dentro do setTimeout

Isso acontece porque o setTimeout é assíncrono e o código que está fora dele é executado imediatamente, sem esperar os 3 segundos.

Interrompendo setTimeout e setInterval

Ambas as funções geram IDs únicos que podem ser utilizados para interromper a execução delas.

Para interromper:

setTimeout: Usamos a função clearTimeout.

setInterval: Usamos a função clearInterval.

Interrompendo setTimeout

Para interromper o setTimeout, precisamos armazenar o retorno do setTimeout em uma variável e passar essa variável para o clearTimeout:

```
let timeoutId = setTimeout(function () {
    console.log('Isso não será executado!');
}, 3000);
clearTimeout(timeoutId);
```

Se rodarmos esse código, o clearTimeout impede que a mensagem seja exibida, mesmo que o tempo de 3 segundos tenha passado.

Interrompendo setInterval

Para interromper o setInterval, fazemos algo similar, mas usamos a função clearInterval:

```
let intervalId = setInterval(function () {
    console.log('Isso seria executado a cada 2 segundos');
}, 2000);
setTimeout(function () {
    clearInterval(intervalId);
    console.log('Intervalo interrompido');
}, 5000);
```

Nesse exemplo, o setInterval é interrompido após 5 segundos, então a mensagem "Intervalo interrompido" será exibida e o setInterval parará de ser executado.

Resumo das Funções

setTimeout:

Executa um código uma única vez após um tempo especificado.

Usado para criar atrasos na execução do código.

Interrompido com clearTimeout.

setInterval:

Executa um código repetidamente em intervalos definidos.

Usado para tarefas repetitivas (por exemplo, atualizar o relógio na tela).

Interrompido com clearInterval.

Conclusão

Nessa aula, vimos como usar as funções setTimeout e setInterval para controlar a execução do nosso código com base em intervalos de tempo. São funções extremamente úteis para controlar eventos assíncronos e são muito utilizadas no desenvolvimento de interfaces de usuário e no controle de fluxos de execução em projetos JavaScript.

Na próxima aula, vamos explorar em mais detalhes a diferença entre código síncrono e código assíncrono, que é um conceito fundamental para entender o comportamento do JavaScript.
