08

Para saber mais: Promise.All

Já sabemos como o callback hell dificulta nossa vida quando estamos falando de manutenção e complexidade de código, como nesse exemplo abaixo, onde temos várias funções auxiliares uma dentro da outra para executar o movimento de um personagem:

```
movePersonagem('100', 'Esquerda', function() {
    movePersonagem('800', 'Direita', function() {
        movePersonagem('200', 'Esquerda', function() {
            movePersonagem('10', 'Direita', function() {
                movePersonagem('60', 'Esquerda', function() {
                })
            })
        })
    })
})
```

COPIAR CÓDIGO

O callback hell acaba sendo necessário nesse caso pois desse modo garantimos que a segunda função só vai ser disparada quando a primeira for concluída e assim por diante, já que nesse cenário de exemplo estamos trabalhando com funções assíncronas para movimentar o personagem.

Vimos também que uma alternativa para deixar o código mais "limpo" é utilizar uma **Promise**. Com o retorno de um objeto de promessa, podemos encadear o `.then()` garantindo a sequência da execução.

```
movePersonagem('100', 'Esquerda')
    .then(() => movePersonagem('800', 'Direita'))
    .then(() => movePersonagem('200', 'Esquerda'))
    .then(() => movePersonagem('10', 'Direita' ))
    .then(() => movePersonagem('60', 'Esquerda' ))
```

COPIAR CÓDIGO

Esse cenário onde fazemos várias requisições que dependem uma da outra é bem comum, e nesse cenário podemos fazer uso do método `.all` da **Promise** .

`Passando cada uma das funções dentro de um array como argumento da` Promise.all `, conseguimos executar todas as funções em ordem sem precisar encadear vários` .then()`

```
Promise.all([
    movePersonagem('100', 'Esquerda'),
    movePersonagem('800', 'Direita'),
    movePersonagem('200', 'Esquerda'),
    movePersonagem(10, 'Esquerda'),
    movePersonagem('60', 'Esquerda')
])
.then(...)
```

COPIAR CÓDIGO

O `Promise.all` vai executar todas as chamadas na ordem e devolver uma resposta que então poderá ser utilizada no `.then` .