Vamos criar um exercício prático para entender melhor o conceito de *callbacks* em *JavaScript*. Suponha que temos uma função baixarConteudo que simula o *download* de um conteúdo da internet. Esta função recebe dois parâmetros: o nome do conteúdo e uma função de *callback* que será chamada quando o *download* estiver concluído.

```
JS callbacks.js > ...
1   function baixarConteudo(nomeConteudo, callback) {
2       console.log(`Iniciando o download de ${nomeConteudo}...`);
3       // Simula um tempo de download (em milissegundos)
4       setTimeout(function () {
5         console.log(`${nomeConteudo} foi baixado com sucesso.`);
6         callback(null, nomeConteudo); // Indica que o download foi concluído com sucesso
7       }, 2000); // Simula 2 segundos de download
8     }
9
10    // Vamos criar uma função callback simples que será chamada após o download
11    function callbackConcluirDownload(erro, nomeConteudo) {
12      if (erro) {
13        console.error(`Erro ao baixar ${nomeConteudo}: ${erro}`);
14      } else {
15        console.log(`Download de ${nomeConteudo} concluído com sucesso!`);
16      }
17    }
18
19    // Agora, vamos usar a função baixarConteudo com callbacks
20    baixarConteudo('Documento.pdf', callbackConcluirDownload);
21    baixarConteudo('Imagem.jpg', callbackConcluirDownload);
22    baixarConteudo('Video.mp4', callbackConcluirDownload);
23
```

Neste exemplo, a função baixarConteudo simula o *download* de um conteúdo e chama a função de *callback* (callbackConcluirDownload) quando o download é concluído. O uso de *callbacks* é comum em operações assíncronas, como *download* de arquivos, chamadas de API, etc.

Observe que a função baixarConteudo recebe o nome do conteúdo e a função de *callback* como parâmetros. Após o término do download simulado, a função de *callback* é chamada, indicando que o download foi concluído. Este é um exemplo básico para ilustrar como os *callbacks* podem ser utilizados em situações assíncronas em *JavaScript*.