# 

## **Material Complementar sobre Callbacks**

As funções JavaScript são executadas na sequência em que são chamadas. Não na sequência em que são definidas.

Este exemplo vai acabar mostrando "Olá" e"Adeus":

### **Exemplo**

|  |
| --- |
| function myFirst() {  console.log("olá"); }  function mySecond() {  console.log("Adeus"); }  myFirst(); mySecond(); |

Este exemplo vai acabar mostrando "Adeus" e "Olá":

### **Exemplo**

|  |
| --- |
| function myFirst() {  console.log("olá"); }  function mySecond() {  console.log("Adeus"); }  mySecond(); myFirst(); |

## **Controle de sequência**

Às vezes você gostaria de ter melhor controle sobre quando executar uma função.   
Suponha que você queira fazer um cálculo e, em seguida, exibir o resultado.

Você pode chamar uma função calculadora (myCalculator), salvar o resultado e, em seguida, chamar outra função (myDisplayer) para exibir o resultado:

### **Exemplo 1**

|  |
| --- |
| function myDisplayer(some) {  console.log(some); }  function myCalculator(num1, num2) {  let sum = num1 + num2;  return sum; }  let result = myCalculator(5, 5); myDisplayer(result); |

Ou, você pode chamar uma função calculadora (myCalculator) e deixar a função calculadora chamar a função de exibição (myDisplayer):

### **Exemplo 2**

|  |
| --- |
| function myDisplayer(some) {  console.log(some); }  function myCalculator(num1, num2) {  let sum = num1 + num2;  myDisplayer(sum); }  myCalculator(5, 5); |

O problema com o **exemplo 1** é que você deve chamar duas funções para exibir o resultado.

O problema com o **exemplo 2** é que você não pode impedir que a função myCalculator exiba o resultado. E além disso, a função myCalculator **DEPENDE** do retorno da função myDisplayer, uma vez que a segunda (myDisplayer) é invocada dentro da primeira (myCalculator).

**Agora é hora dos callbacks:**

## **Callback JavaScript**

Um callback é uma função que é passada como um argumento para outra função.   
Esta técnica permite que uma função chame outra função.

Com um callback, você pode chamar a função calculadora (myCalculator) com um callback (myCallback) e deixar a função myCalculator executar o callback somente após o cálculo ser concluído:

### **Exemplo:**

|  |
| --- |
| function myDisplayer(some) {  console.log(some); }  function myCalculator(num1, num2, myCallback) {  let sum = num1 + num2;  myCallback(sum); }  myCalculator(5, 5, myDisplayer); |

No exemplo acima, myDisplayer é o nome de uma função.

Ela foi passada para o myCalculator() como argumento.

Ao passar uma função como argumento, lembre-se de não usar parênteses.

**CORRETO:** myCalculator(5, 5, myDisplayer);

**ERRADO:** myCalculator(5, 5, myDisplayer());

## 

## **Quando usar callback?**

Os exemplos acima foram simplificados para ensinar a você a sintaxe de um callback.

O local que os callbacks realmente brilham é em funções assíncronas, onde uma função tem que esperar por outra função (como esperar um arquivo para carregar).

Ansioso para isso, certo?  
  
As funções assíncronas serão abordadas no próximo capítulo!

Referências: <https://javascript.info/callbacks>