CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DO ESTADO DE MINAS GERAIS

Caio Henrique Noronha

Gustavo Soares Silva

**Atividade em Dupla**

*Callbacks* e *Promises*

Divinópolis

2019

De forma simples, *callback* é uma função passada como parâmetro para outra função. A função setTimeout recebe dois parâmetros: o primeiro é uma função *callback*, e o segundo é o tempo que o interpretador irá esperar até executar essa função.

*Callbacks* são funções que serão executadas de modo assíncrono, ou posteriormente. Ao passo que o código for lido de cima para baixo de modo processual, programas assíncronos possivelmente executam funções em tempos diferentes baseado na ordem e velocidade em que requisições http ou o trabalho de arquivamento do sistema acontecem.

Na programação assíncrona, callbacks são passadas como funções para serem executadas após um certo evento. Por exemplo, um programa calcula o salário líquido a partir do bruto e faz algo com esse valor calculado. Nesse caso, podemos fazer da seguinte forma:

------------------------------------------------------------------------------------------------

1 let salarioBruto = 3000;  
2 let salarioLiquido;getSalario(salarioBruto, (resultado) => {  
3 salarioLiquido = resultado;  
4 console.log(`O salário liquido é ${salarioLiquido}`);  
5 });function getSalario(salarioBruto, callback)  
6 {  
7 let liquido = 0; const inss = salarioBruto \* 0.11;  
8 const vr = salarioBruto \* 0.05;  
9 const vt = salarioBruto \* 0.06;  
10 const fgts = salarioBruto \* 0.15; const descontos = inss + vr + vt + fgts; liquido = 11 salarioBruto - descontos;  
12   
13 return callback(liquido);  
14 }

------------------------------------------------------------------------------------------------

Exemplo 1.

A nossa *callback* é definida na passagem do segundo parâmetro durante a invocação da getSalario: **getSalario(salarioBruto, (resultado) => …**  
Na declaração da nossa função *callback*, estamos dizendo que ela irá receber um parâmetro que é o resultado do cálculo do salário líquido, e o código que ela irá executar será uma atribuição e depois um console.log, informando o salário líquido.

Outro exemplo. Supondo que precisamos armazenar nosso numero em um arquivo chamado number.txt:

----------------------------------------------------------------------------------------------------------

1 var fs = require('fs') // require é uma função especial do node para requisitar os módulos

2 var myNumber = undefined // não sabemos ainda qual valor esta armazenado no arquivo

3

4 function addOne() {

5 fs.readFile('./number.txt', function doneReading(err, fileContents) {

6 myNumber = parseInt(fileContents)

7 myNumber++

8 })

9 }

10

11 addOne()

12

13 console.log(myNumber) // mostra na saida padrão `undefined`

----------------------------------------------------------------------------------------------------------

Exemplo 2.

Em outro exemplo, pense no restaurante do exemplo do tutorial lá do início. Em muitos restaurantes você pega um número e coloca em sua mesa e espera por sua comida. Eles são como os *callbacks*. Isso deixa claro para o servidor a quem chamar quando o *cheeseburger* estiver pronto.

Colocando console.log em uma função e passando ele para um *callback*.

------------------------------------------------------------------------------------------

1 var fs = require('fs')

2 var myNumber = undefined

3 function addOne(callback) {

4 fs.readFile('./number.txt', function doneReading(err, fileContents) {

5 myNumber = parseInt(fileContents)

6 myNumber++

7 callback()

8 }

9 }

10

11 function logMyNumber() {

12 console.log(myNumber)

13 }

14

15 addOne(logMyNumber)

------------------------------------------------------------------------------------------

Exemplo 3.

Além disso, no JavaScript existem *Promises*. Uma [*Promise*](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Promise) é um objeto que representa a eventual conclusão ou falha de uma operação assíncrona. Essencialmente, uma *promise* é um objeto retornado para o qual você adiciona *callbacks*, em vez de passar *callbacks* para uma função.

Utilizando os exemplos anteriores, pode-se modifica-los para utilizar promises.

Exemplo 1:

------------------------------------------------------------------------------------------------

1 let salarioBruto = 5000;

2 let salarioLiquido;

3 promiseOk = new Promise((resolve, reject) => {

4 getSalario(salarioBruto, (resultado) => {

5 salarioLiquido = resultado;

6 resolve(`O salário liquido é ${salarioLiquido}`);

7 });

8 })

9 function getSalario(salarioBruto, callback){

10 let liquido = 0;

11 const inss = salarioBruto \* 0.11;

12 const vr = salarioBruto \* 0.05;

13 const vt = salarioBruto \* 0.06;

14 const fgts = salarioBruto \* 0.15;

15 const descontos = inss + vr + vt + fgts;

16 liquido = salarioBruto - descontos;

17

18 return callback(liquido);

19 }

20 promiseOk .then((r) => {

21 console.log(r)

22 })

------------------------------------------------------------------------------------------------