FUNÇÕES
(AULA 6)

CURSO BÁSICO DE PROGRAMAÇÃO COM JAVASCRIPT

MAYARA MARQUES

mmrosatab@gmail.com

JS

SUMÁRIO

- Funções
- Estrutura de uma função
- Função retornando função
- Função com parâmetro default
- Função anônima
- Função de callback
- Mão na massa

FUNÇÕES

Funções são blocos de código que **realizam uma tarefa específica** e que **podem ser chamados em outras partes do código**.

FUNÇÕES

O uso de funções proporciona:

- Reutilização de código
  - O que proporciona um menor número de linhas de código.
- Organização
  - Por consequência, uma melhor legibilidade do código.
- Modularização
  - Isso significa que haverá mais facilidade em fazer manutenção em uma parte do código sem afetar outras.

ESTRUTURA DE UMA FUNÇÃO

Palavra reservada que indica a declaração de uma função.

Parâmetros da função.
Representam as variáveis que assumirão valores quando a função for chamada/invocada.
Permitem utilizar valores que estão fora da função.

```
function name(parameter1, parameter2, parameter3) {

    // code to be executed

}
```

Nome da função

Conteúdo interno da função, onde ficarão as instruções que se quer executar quando a função for chamada.

ESTRUTURA DE UMA FUNÇÃO

As funções podem apresentar retorno. O retorno é um valor que é especificado dentro da função para ser retornado.

```
function name(parameter1, parameter2, parameter3)
{
    // code to be executed
    return value
}
```

Valor a ser retornado

Palavra reservada utilizada para fazer o retorno

ESTRUTURA DE UMA FUNÇÃO

```
function sum(num1, num2) {
    return num1 + num2
}


sum(3,4)


console.log(sum(3,4)) // 7
```

Para invocar/chamar uma função, use seu nome e, em parênteses, os seus argumentos.

ESTRUTURA DE UMA FUNÇÃO

O retorno de uma função pode ser de qualquer tipo de dados (string, number, bool, array, objeto e etc).

```
function hello() {
    return "Hello World"
}
```

Para invocar/chamar uma função, use seu nome e, em parênteses, os seus **argumentos**.

```
hello()
```

```
console.log(hello()) // Hello World
```

ESTRUTURA DE UMA FUNÇÃO

O retorno de uma função pode ser de qualquer tipo de dados (string, number, bool, array, objeto e etc).

```
function hello() {
    return "Hello World"
}
```

Para invocar/chamar uma função, use seu nome e, em parênteses, os seus **argumentos**.

```
hello()
```

```
console.log(hello()) // Hello World
```

ESTRUTURA DE UMA FUNÇÃO

Não confundir parâmetro com argumento!

É a variável definida na declaração da função.
'name' é um **PARÂMETRO**

```
function greeting(name) {
    console.log(`Hi, ${name}!`)
}

greeting("Maria") // Hi, Maria!
```

Nessa chamada de função, a variável 'name' assumirá o valor 'Maria'.

É o valor passado para a função quando ela é chamada. **Dentro** da função será representado por 'name'.

ESTRUTURA DE UMA FUNÇÃO

Não confundir parâmetro com argumento!

'name' aqui é um **PARÂMETRO**

```
function showName(name) {
    console.log(`The name is: ${name}`)
}

const name = "Carlos"

showName(name) // The name is Carlos
```

'name' aqui é uma **VARIÁVEL**

'name' como **ARGUMENTO**
O **valor da variável** 'name' será atribuído ao **parâmetro** 'name' que foi definido na declaração da função showName.

Embora a variável e o parâmetro sejam nomeados com **'name'.** Eles são coisas **DISTINTAS**!

ESTRUTURA DE UMA FUNÇÃO

Não confundir parâmetro com argumento!

'name' aqui é um **PARÂMETRO** e o conteúdo de 'name' será 'Carlos' quando a função for invocada na linha mais abaixo.

```
function showName(name) {
    console.log(`The name is: ${name}`)
}

const personName = "Carlos"

showName(personName) // The name is Carlos
```

'personName' aqui é uma **VARIÁVEL**

'personName' sendo passado como **ARGUMENTO** da função showName

ESTRUTURA DE UMA FUNÇÃO

Não confundir parâmetro com argumento!

'name' aqui é um **PARÂMETRO**

```
function showName(name) {
    console.log(`The name is: ${name}`)
}


showName("Carlos") // The name is Carlos
showName("Maria") // The name is Maria
showName("Joana") // The name is Joana
```

Função sendo chamada com argumentos diferentes.

Parâmetro → Definido na função, uma espécie de "espaço reservado" para o valor.
Argumento → Valor real atribuído ao chamar a função.

Parâmetro = O que a função espera.
Argumento = O que é realmente passado para a função.

ESTRUTURA DE UMA FUNÇÃO

Quando uma função não possui retorno explícito ou apresenta a palavra reservada return seguida de ponto e vírgula (return ;), seu retorno será undefined.

```
function noReturn(){}

console.log(noReturn()) // undefined

function returnWithSemicolon(){
    return
}

console.log(returnWithSemicolon()) // undefined
```

FUNÇÃO RETORNANDO FUNÇÃO

O retorno de uma função pode ser de qualquer tipo de dados. Em javascript, pode-se também retornar uma função.

FUNÇÃO RETORNANDO FUNÇÃO

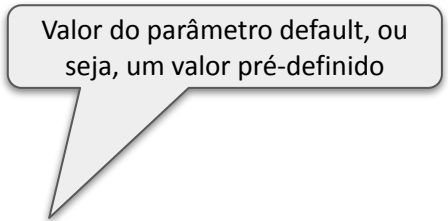
```
function calculator(op) {
  if (op === "+") {
    return function(a, b) {
      return a + b
    }
  } else if (op === "-") {
    return function(a, b) {
      return a - b
    }
  }
}
```

```
const sum = calculator("+")
console.log(sum(2, 3)) // 5

const sub = calculator("-")
console.log(sub(10, 2)) // 8
```

FUNÇÃO COM PARÂMETRO DEFAULT

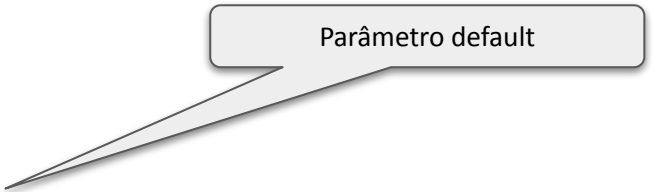
Javascript permite passagem de parâmetros default. Um parâmetro default é um parâmetro que possui um valor que é utilizado dentro da função sempre que seu respectivo argumento não for passado. Isso significa que em caso de omissão de um valor no momento da chamada da função, o valor do parâmetro default será utilizado.

FUNÇÃO COM PARÂMETRO DEFAULT




```
function name(parameter1, parameter2=value) {
    // code to be executed
    return value
}
```

FUNÇÃO COM PARÂMETRO DEFAULT




```
function increment(num, step=1) {
    return num + step
}


console.log(increment(9)) // 10
console.log(increment(9, 2)) // 11
```

Função anônima

Uma função pode ser criada sem um nome. Quando isso acontece, esta é chamada de função anônima.

```
(function() {
    console.log("I'm an anonymous function")
    console.log("Look at me! I don't have name")
})
```

FUNÇÃO ANÔNIMA

Para executar uma função anônima, existem duas opções:

1. Executá-la logo após sua criação com o símbolo `()`

```
(function() {
    console.log("I'm an anonymous function")
    console.log("Look at me! I don't have name")
}())


// "I'm an anonymous function"

// "Look at me! I don't have name"
```

FUNÇÃO ANÔNIMA

JS

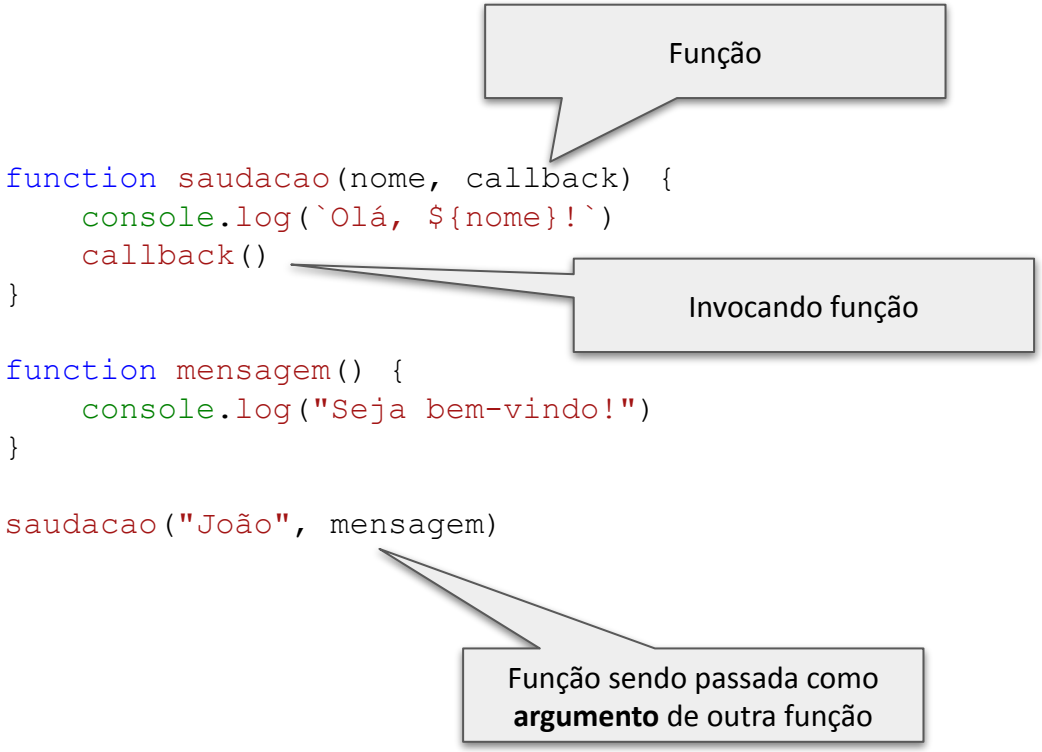
2. Atribuir a função a uma variável e tratar a variável como se fosse a função fazendo uma invocação com o parênteses.

```
let anonymous = function() {
    console.log("I'm an anonymous function")
    console.log("Look at me! I don't have name")
}

anonymous()
// "I'm an anonymous function"

// "Look at me! I don't have name"
```

FUNÇÃO DE CALLBACK

Funções podem ser passadas como parâmetro de outra função e, mais tarde, serem executadas dentro de função que a recebeu. Quando esse processo ocorre, a função passada como parâmetro é chamada de função de callback .

FUNÇÃO DE CALLBACK



MÃO NA MASSA

MÃO NA MASSA

1. Crie um programa que calcule a média aritmética de três notas digitadas por meio de uma função chamada média.

2. Cria uma função que retorne verdadeiro ou falso para informar se um número é primo ou não.

3. Cria uma função que calcule o fatorial de um número.

4. Modifique o código da função **`calculator` (slide Função retornando função)** para disponibilizar também a função de multiplicar(*) e a função de dividir(/).

5. Crie uma função que receba como parâmetro nome, telefone e e-mail e retorne uma frase qualquer com essas informações.

MÃO NA MASSA

6. Crie uma função que receba dois números, base e expoente, e retorne o resultado da base elevado ao expoente.

7. Modifique o programa anterior para que o expoente seja o número 2 em caso de omissão do expoente no momento da chamada função.

8. Crie uma função que receba dois números, intervalo inferior e intervalo superior, e retorne o somatório dos números contidos entre eles. Sua função deve retornar -1, caso os números sejam iguais ou o intervalo superior for menor que inferior ou o intervalo inferior for maior que o intervalo superior.

9. Crie um programa que realiza o somatório dos números digitados pelo usuário e, ao final, exiba o somatório dos números coletados. Seu programa só deve parar de coletar os números quando o usuário digitar a palavra “PARAR”. Utilize função!

10. Elabore uma função que recebe um nome, uma idade e uma função de callback. Ao executar a função, deve ser impressa a mensagem “Olá, meu nome é *‘nome’* e tenho “*idade*” anos!”. Utilize a função de callback para montar a mensagem.

11. Crie uma função anônima que imprima “Hello World!”.

REFERÊNCIAS

- https://developer.mozilla.org/pt-BR/docs/Glossary/Callback_function
- https://www.w3schools.com/js/js_function_definition.asp
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Working_with_objects