Sumário

[**Aula 01.** **Tipos de função** 2](#_Toc93587888)

[Função Anônima: 2](#_Toc93587889)

[Função Autoinvocável: (IIFE – Immediately Invoked Function Expression) 2](#_Toc93587890)

[Função Callbacks 3](#_Toc93587891)

[Objeto “Arguments” 4](#_Toc93587892)

[**Aula 02.** **Parâmetros** 5](#_Toc93587893)

[**Aula 03.** **Loops** 6](#_Toc93587894)

[**Aula 04.** **A palavra this** 7](#_Toc93587895)

[**Aula 05.** **Arrow functions** 8](#_Toc93587896)

[**Aula 06.** **Atividade Prática** 9](#_Toc93587897)

# **Tipos de função**

**Sintaxe básica de uma função:**

Function nome (parâmetros){

//instruções

return; // valor a ser retornado

}

## Função Anônima:

São funções que representam expressões.

Uma variável pode armazenar uma função.

const soma = function (a,b){

return a+b;

}

soma(1,2); // retorna (3)

soma(5,4); // retorna (9)

## Função Autoinvocável: (IIFE – Immediately Invoked Function Expression)

Uma função anônima entre parênteses, seguida por outro par de parênteses, que representa sua chamada.

**Exemplo 01:**

**(**

**function() {**

**let name = “Digital Innovation One”;**

**return name;**

**}**

**)();**

**Exemplo 02:**

**(**

**function(a,b) {**

**return a+b;**

**}**

**)(1,2);**

Também pode ser utilizada com parâmetros ou armazenada em uma variável.

**const soma3 = (**

**function(a,b) {**

**return a+b;**

**}**

**)(1,2);**

**console.log(soma3);**

## Função Callbacks

Uma função Callback passada como argumento para outra.

**const calc = function(operação, num1, num2){**

**return operação(num1, num2);**

**}**

**const soma = function(num1, num2){**

**return num1 + num2;**

**}**

**const subtração = function(num1, num2){**

**return num1 - num2;**

**}**

**const resultSoma = calc(soma,1,2);**

**const resultSub = calc(subtração,1,2);**

**console.log(resultSoma);**

**console.log(resultSub);**

# **Parâmetros**

## Objeto “Arguments”

Um array com todos os parâmetros passados quando a função foi invocada.

**function findMax(){**

**let max = -Infinity;**

**for(let i = 0; i< arguments.length; i++){**

**if (arguments[i] > max){**

**max = arguments[i];**

**}**

**}**

**return max;**

**}**

**console.log(findMax(100,150,1,2,3,6,90,180));**

## Arrays

**Spread:** uma forma de lidar separadamente com elementos, representado por “...”.

O que era parte de um array se torna um elemento independente.

**function sum(x,y,z){**

**return x + y + z;**

**}**

**const numbers = [1,2,3];**

**console.log(sum(...numbers));**

**Rest:** combina elementos em um array, representado por “...”.

O que era um elemento independente se torna parte de um array.

**function confereTamanho(...args){**

**console.log(args.length);**

**}**

**confereTamanho(); //0**

**confereTamanho(1, 2); //2**

**confereTamanho(3, 4, 5); //3**

## Objetos

Object Destructuring

Entre chaves {}, podemos filtrar apenas os dados que nos interessam em um objeto.

**const user = {**

**id:42,**

**displayName: 'jdoe',**

**fullName: {**

**firstName: 'John',**

**lastName: 'Doe'**

**}**

**};**

**function userId({id}){**

**return id;**

**}**

**function getFullName({fullName: {firstname: first, lastName: last}}){**

**return `${first} ${last}`;**

**}**

**userId(user); //42**

**getFullName(user); // John Doe**

# **Loops**

# **A palavra this**

A palavra reservada **this** é uma referência de contexto.

No exemplo, **this** refere-se ao objeto pessoa.

**const pessoa = {**

**firstName: ‘André’,**

**lastName: ‘Soares’,**

**id: 1,**

**fullName: function(){**

**return this.firstName + ‘ ‘ + this.lastName;**

**},**

**getId: function(){**

**return this.id;**

**}**

**}**

**pessoa.fullName();**

**pessoa.getId();**

Seu valor pode mudar de acordo com o lugar no código onde foi chamada.

|  |  |
| --- | --- |
| **Contexto** | Referência |
| Em um objeto (método) | Próprio objeto dono do método |
| Sozinha | Objeto global (em navegadores, window) |
| Função | Objeto global |
| Evento | Elemento que recebeu o evento |

## Método Call e Apply

O método Call permite com que façamos a chamada de uma função onde passamos qual objeto fará referência e dentro dela, dizemos qual o campo que estaremos pegando.

const pessoa = {

nome : 'Leonardo',

};

const animal = {

nome : 'Betina',

};

function getSomething(){

console.log('dentro do getSomething');

console.log(this.nome);

}

console.log('antes do call');

getSomething.call(pessoa);

Podemos passar parâmetros separados por vírgula também

const myObj ={

num1:2,

num2:4,

}

function soma(a,b){

console.log(this.num1 + this.num2 + a + b);

}

soma.call(myObj,1,5); // se eu não colocar o myObj, dá erro uma vez que é necessário para que a Call funcione

A diferença entre o **Call** e o **Apply**, é que no lugar do **soma.call(myObj,1,5);**, utilizaremos o **soma.call(myObj,[1,5]);**, utilizanos um array.

## Método Bind

O método **bind()** cria uma nova função que, quando chamada, tem sua palavra-chave this definida com o valor fornecido, com uma sequência determinada de argumentos precedendo quaisquer outros que sejam fornecidos quando a nova função é chamada.

A função bind() cria uma nova **função vinculada** (bound function). Uma função vinculada é um **objeto de função exótico** (termo da **ECMAScript 2015**) que encapsula o objeto de função original. Chamar uma função vinculada geralmente resulta na execução de sua **função encapsulada**.

**Exemplo**:

this.x = 9; //this aqui se refere ao objeto global "window" do navegador

var module = {

x: 81,

getX: function() {

console.log('dentro do function getx');

return this.x;

},

y: 32,

gety: function() {

console.log('dentro do function gety');

return this.y;

}

};

// Retornará o que está dento do módulo, só pelo simples fato de ter os ()

console.log(module.getX()); // 81

console.log(module.gety()); // 32

// Retornará o que está fora do módulo, só pelo simples fato de não ter os (), para pegar o valor de dentro do módulo, temos que colocar o .bind igual ao que está no final.

var retrieveX = module.getX;

console.log(retrieveX()); // 9

var retrieveY = module.gety;

console.log(retrieveY()); //undefined pois não tenho nenhum this.y atribuído fora do módulo

// retorna 9 - a função foi invocada no escopo global

// Criando uma nova função com 'this' vinculada ao módulo

// Programadores novatos podem confundir a variável x

// global com a propriedade x do módulo

var boundGetX = retrieveX.bind(module);

console.log(boundGetX()); // 81

# **Arrow functions**

# **Atividade Prática**