FUNÇÕES

1. TIPOS DE FUNÇÃO

* Estrutura –É utilizada a palavra reservada function, e segue da seguinte forma: function nome (parâmetros) {

} Obs: Qualquer variável criada dentro de uma função apenas podem ser utilizadas dentro dela.

* Return – Quando invocamos a palavra “return”, a função para de ser executada, e ele retorna o valor que vem após ele, você pode ou não declarar um valor
* Função Anônima – Uma variável pode armazenar uma função, nesse caso a função ela não tem um nome apenas os parâmetros, EX:

Const soma = function (a, b) { return a + b;}

Soma(1, 2) – vai retornar 3.

* Função autoinvocável e Callbacks – Conhecida pela expressão IIFE (immediately invoked function expression), é realizada uma função anônima entre parênteses, seguida por outro par de parênteses, que representa sua chamada. EX : (

function () {

let name = “Marcelo”

return name; }

)();

Também podem ser utilizadas com parâmetros ou ser armazenadas em uma variável.

* Callbacks – Uma função passada como argumento para outra. Utilizando callbacks, você tem maior controle da ordem de chamadas. EX:

Const calc = function (operação, num1, num2) { return operação (num1, num2); }

Const soma = function (num1, num2) { return num1 + num2; }

Const sub = function (num1, num2) { return num1 - num2; }

Const resultSoma = calc(soma, 1, 2);

Const resultSub = calc(sub, 1, 2);

Console.log(resultSub); // -1

Console.log(resultSoma); // 3

1. PARÂMETROS

* Valores padrão – Atualmente POS -ES2015: Ao você escrever um parâmetro você atribui a o paramento um valor, caso esse parâmetro não seja definido ela terá aquele valor. EX:

Function exponencial(array, num = 1) { const result = []; …

* Objeto arguments – Basicamente é um array uma coleção com todos os parâmetros que foram passados quando a função foi invocada palavra reservada arguments
* Array e Objetos – Quando se utiliza o spread (...) o que era uma parte do array se torna um elemento independente
* Spread – É uma forma de lidar separadamente com os elementos
* Rest – Combina os elementos em um array, ou seja, o que era um elemento independente se torna um array.
* Object Destructuring – Entre chaves {}, Podemos filtrar apenas os dados que nos interessam em um objeto. EX:

Const user = {

Id: 42

displayName: ‘jdoe’

fullName: {

firstName: ‘John’

lastName: ‘doe’

}};

Function userId ({id}) {

return id; }

userId (user)

// 42

1. LOOPS

* If/else e switch – De acordo com as boas práticas é recomendado você crias sua condição dentro de uma variável e associar ao if. EX:

function numeroPositivo(num) {

const ehNegativo = num < 0;

if (ehNegativo) { return false;}

return true }

* For – É um loop dentro de elementos iteráveis. EX :

function multiplicaPorDois (arr) {

let multiplicados = []

for(let I = 0; I <arr.length; i++ {

multiplicados.push(arr[i] \* 2);

}

Return multiplicados;

Const meusNumeros = [2, 33, 456, 356, 40];

multiplicaPorDois(meusnumeros);

// 4, 66, 912, 712, 80

* For e In – Loop entre propriedades enumeráveis de um objeto. EX:

Funtcion forInExemplo(obj) {

For(prop in obj) {

Console.log(prop);}}

Const meuObjeto = {

nome: “joão”

idade: “20”

cidade:”salvador” }

forInExemplo(meuObjeto);

// nome, idade, cidade.

OU

Funtcion forInExemplo(obj) {

For(prop in obj) {

Console.log(obj[prop]);}}

Const meuObjeto = {

nome: “joão”

idade: “20”

cidade:”salvador” }

forInExemplo(meuObjeto);

// joão, 20, salvador

* For of – loop entre estruturas iteráveis (array, strings).

Function logLetras(palavra) {

For (letra of palavra) {

Console.log(letra);}}

Const palavra = “abacaxi”;

logLetras(palavra);

//a, b, a, c, a, x, i

* While -Executa uma instrução ate que essa instrução se torne falsa. EX:

Function exemploWhile () {

Let num = 0

While (num <=5) {

Console.log(num);

Num++;}}

exemploWhile ()

// 0, 1, 2, 3, 4, 5

* Do while – Mesmo se a condição seja falsa, ele executará uma vez e dpois irá parar. Ex :
* Function exemploDoWhile () {

Let num = 6;

Do { Console.log(num);

Num++;} while (num <=5) {

exemploDoWhile ()

// 6

1. THIS

* O que é this – Essa palavra reservada é uma referência para o contexto no qual você está, quando você declara o this dentro de um método ele vai se referir ao objeto.

Obs. ( Sempre que uma função está dentro de um objeto ela é um método)

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

|  |  |
| --- | --- |
| CONTEXTO | REFERÊNCIA |
| Em um objeto(método) | Próprio objeto dentro do método |
| sozinha | Objeto global (em navegadores, window) |
| função | Objeto global |
| evento | Elemento que recebeu o evento |

* Manipulando seu valor
  + Call – esse método, você vai chamar um objeto no qual você quer que o this se refira. EX:

Const pessoa = {

Nome: “Miguel”};

Const animal = {

Nome: “Murphy”};

Function getSomething () {

Console.log (this.nome);}

getSomething.call(pessoa);

//Miguel

È possível usar essa chamada do call, quando a função tem argumentos (parâmetros) EX:

const myObj = {

num1: 2,

num2: 4,

};

Function soma (a, b) {

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

}

Soma.call (myObj, 1, 5);

//12

* Apply – Funciona parecido com o call, mas tem uma pequena diferença. EX:

Const pessoa = {

Nome: “Miguel”,};

Const animal = {

Nome: “Murphy”};

Function getSomething () {

Console.log (this.nome);}

getSomething.apply(pessoa);

//Miguel

A diferença é que quando você for utilizar dentro de argumentos, os argumentos devem ser declarados dentro de um array. EX:

const myObj = {

num1: 2,

num2: 4,

};

Function soma (a, b) {

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

}

Soma.apply (myObj, [1, 5]);

//12

* Bind – Clona a estrutura da sua função onde foi chamado, e vai aplicar o valor do objeto passado como parâmetro. EX:

Const retornaNomes = function () {

Return this.name;};

let bruno = retornaNomes.bind ({name: “Bruno”});

bruno();

//Bruno

1. ARROW FUNCTION

* Sintaxe – è representada por esse símbolo =>, quando a sua função tiver apenas 1 linha você pode utilizar o arrow function ao invés do return, está sempre associada a uma variável. EX:

Const helloWorld = () => “Hello World”;

Obs: 1 - Caso exista apenas uma linha, pode dispensar as chaves e return.

2- Caso exista apenas um parâmetro pode dispensar os parênteses.

EX:

Const soma = (a, b) => a + b;

Soma (4, 6);

//10

Ou

Const soma = a => a;

Soma (5);

//5

ARROW FUNCTION NÃO FAZ HOISTING.

* Outras restrições – os métodos do This não vai funcionar (bind, call, aplly), não existe o objeto arguments, o construtor (ex new MeuObjeto()) também não pode ser utilizado