* **TIPOS DE FUNÇÕES**

**APRESENTAÇÃO DO CURSO**

Objetivos

1. Mostrar como declarar funções
2. Como manipular parâmetros
3. Apresentar loops e outras declarações
4. Apresentar o argumento “this”

**ESTRUTURA E FUNÇÃO ANÔNIMA**

Estrutura = Definição comum de uma função

* Function nome (parametros) {

//instruções

Return; //valor de retorno

}

* Função Anonima = Funções que representam expressões

Const soma = function (a, b) {

Return a + b;

}

Soma(1, 2) // 3

Soma(5, 4) // 9

**FUNÇÃO AUTOINVOCAVEL E CALLBACKS**

* Função Autoinvocavel

Conhecida por IIFE (Immmediately Invoked Function Expression)

Essas funções não podem receber parâmetros também

(

Function( ) {

Let name = “Digital Innovation One”

Return name;

}

)( );

// Digital Innovation One

* Callbacks

Uma função passada como argumento para outra

Const calc = function(operação, num1, num2){

Return operação(num1, num2);

}

Const soma = function(operação, num1, num2){

Return operação(num1 + num2);

}

Const sub = function(operação, num1, num2){

Return operação(num1 - num2);

}

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

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

Console.log(resultSub); //-1

Console.log(resultSoma); //3

* **PARAMETROS**

VALORES PADRÃO E OBJETO “ARGUMENTS”

* Valor padrão

Function exponencial(array, num =1) {

Const result = [ ];

For (let i = 0; i < array.lenght; i++) {

Result.push(array[i] \*\* num);

}

Return result;

}

Exponencial ([1, 2, 3, 4])

// [1, 2, 3, 4]

Exponencial ([1, 2, 3, 4], 4)

// [1, 8, 27, 64]

* Objeto “Arguments”

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

ARRAYS e OBJETOS

Spread (técnica): Uma forma de lidar separadamente com elementos;

Oque 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 (técnica): Combina os argumentos em um array;

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

Object Destructuring

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

* **LOOPS**

IF/ELSE e SWITCH

* If/else

If = Declaração 1 (dentro de if): Ocorre caso a condição seja verdadeira.

Else = Declaração 2 (dentro de else): Ocorre caso a condição seja falsa.

Lembrando que JavaScript não tem Elseif

* Switch

Equivale a uma comparação de tipo e valor (===)

Sempre precisa de um valor “default”

Ideal para quando se precisa comparar muitos valores

FOR e WHILE

* for

Loop dentro de elementos iteráveis (arrays, strings)

* **THIS**

O QUE É THIS

A palavra reservada this é uma referência de contexto.

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) |
| Evento | Objeto global |
| Evento | Elemento que recebeu o evento |

MANIPULANDO SEU VALOR

* Call
* Apply

Deve usar [ ] array para essa função

* Bind

Clona a estrutura da função onde é chamada e aplica o valor do objeto passado como parâmetro.

* **ARROW FUNCTIONS**

SINTAXE e OUTRAS RESTRIÇÕES

Arrow functions - =>

ATIVIDADE PRÁTICA

**Qual o valor de this?**

Depende do contexto no qual a palavra é invocada

##### O que são callbacks?

Funcões passadas como parâmetros para outras funções

##### Dado o código for(let i = 0; i < array.length; i++) { console.log(array[i]) } Qual a afirmativa correta?

A partir do índice 0, enquanto o índice for menor que o tamanho do array, o código irá logar o elemento que estiver em cada índice do array

##### Sobre o método bind, é correto afirmar que

Cria um clone da função na qual for invocado, designando a this o valor do objeto passado como parâmetro.

##### Sobre funções anônimas, é correto afirmar que

São funções sem um nome especifico e que podem ter ser valor atribuído a uma variável

##### Sobre o método apply, é correto afirmar que

Transforma o valor de this no do objeto enviado como primeiro parâmetro e aceita outros parâmetros através de um array. É reutilizável em outros objetos

##### Sobre funções autoinvocáveis, assinale a alternativa falsa:

Não podem receber parâmetros

##### Sobre o método call, é correto afirmar que

Transforma o valor de this no do objeto enviado como primeiro parâmetro e aceita outros parâmetros separados por vírgulas. É reutilizável em outros objetos

##### Dado o código function minhaFuncao(num = 0) { return num; } É correto afirmar que

A função possui um valor padrão

##### Dadas as afirmações abaixo I. Funções sempre precisam de parâmetros; II. Funções são determinadas pela palavra reservada function; III. Funções podem retornar ou não algum valor; IV. O escopo de uma variável criada dentro de uma função é a própria função. São verdadeiras:

II e III