Tipos de Função

Estrtura

Function nome(parametros) {

// intruções

Return; //Para a função e retorna o valor

}

Função Anônima

Funções que representam expressões

Uma variável armazena uma função

Const soma = function (a, b) {

Return a + b;

}

Soma(1, 2) // 3

Soma(5, 4) //9

Função auto invocável

IIEF

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

(

Function( ) {

Let name = “DIO”

Return name;

}

) ( );

//DIO

Callbacks

Uma função passada como argumento para outra função

Parâmetros

Valores Padrão

Arguments é um array com todos os parâmetros passados quando a função foi invocada

Arrays e Objetos

Arrays

Spread: uma forma de lidar separadamente com elementos

Function sum(x, y, z) {

Return x + y + z;

}

Const numbers = [1, 2, 3]

Console.log(sum(...numbers))

O que é parte de um array se torna um elemento independente

**Rest**: combina os argumentos em um array

Function confereTamanho(...args) {

Console.log(args.length)

}

confereTamanho( ) // 0

confereTamanho( 1, 2) // 2

confereTamanho( 3, 4, 5) // 3

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

Objetos

Desestruturação de Objetos

Entre chaves podemos filtrar apenas os dados que nos interessam em um objeto

Loops

If/else

Declaração 1 (dentro do if): Ocorre caso a condição seja verdadeira

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

Switch/case

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

Sempre precisa de um valor “default” / default ocorre quando o input não cai em nenhum dos valores especificados.

Ideal para quando se precisa comparar muitos valores

For

Loop dentro de elementos iteráveis (arrays, strings)

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

Inicia uma variável; uma condição; e contador.

For...in

Loop entre propriedades enumeráveis de um objeto

For...of

Loop entre estruturas iteráveis (arrays, strings)

While

Executa instruções até que a condição se torne falsa

Do...While

Executa instruções até que a condição se torne falsa

Porém a primeira execução sempre ocorre

This

Sempre que uma função está dentro de um objeto o chamamos de método

A palavra reservada this é uma referência de contexto

Ex. return this.chave e isso ira retornar o valor dessa chave

Seu valor pode mudar de acordo com o lugar no código

Onde foi chamado

|  |  |
| --- | --- |
| Contexto | Referencia |
| Em um objeto (método) | Próprio objeto dono do método |
| Sozinho | Objeto global (em navegadores, window) |
| Função | Objeto global |
| Evento | Elemento que recebe o evento |

Call

Variável.call(x) o x é o valor o qual você quer que o this se refira

Apply

Variável.apply(x) é possível passar parâmetros para essa função dentro de um array

Call manda os valores separado por virgulas enquanto o apply manda eles é um array

Bind

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

Variável.bind({nome: ‘valor’})

Arrow function

Sintaxe

Const função = ( ) => {

Return “Hello world”;

}

Se a sua arrow function for curta/1 linha não precisa das {} nem do return

Const função = ( ) => “Hello world”;

Caso exista apenas um parametro, pode dispensar os ( )

Const função = a => a;

Restrições

Arrow function não faz hoisting

“this” sempre será objeto global. Métodos para modificar seu valor não irão funcionar (call, bind e aplly)

Não existe o objeto “arguments” (junta dentro de uma lista todos os argumentos da função)

O construtor (ex: new MeuObjeto()) também não pode ser utilizado