# Maneiras de declarar funções

* Clássica:

function falaOi(){

    console.log('oi')

}

falaOi()

* Function Expression

const falaOi = function(){

    console.log('oi')

}

falaOi()

* Arrow Function

const falaOi = () =>{

    console.log('oi')

}

falaOi()

# Parâmetros da função

Quando uma função é declarada com *function*, ainda que a função não receba parâmetros, podemos executá-la enviando parâmetros, que ficam armazenados em *arguments*

function parametro() {

    console.log(*arguments*)

}

parametro('ola','mundo','!')

-------------------------

CONSOLE: [Arguments] { '0': 'ola', '1': 'mundo', '2': '!' }

# Retorno da função

O argumento *return* retorna um valor e termina a execução da função.

//retorno de função

function multiplicador(*multiplicador*){

    return function(*n*){

        return *n* \* *multiplicador*

    }

}

const duplica = multiplicador(2) //duplica recebe uma função

                                 //que retornará número vezes multiplicador

//console.log

console.log(duplica(3)) //resultado "6"

# Escopo léxico

Nada mais é do que a função “saber” o que pode se acessar, através do seu escopo e dos escopos próximos.

//Escopo léxico

const num = 1

function contar(){

    console.log(num)

}

function usarContar(){

    const num = 2

    contar()

}

//console.log

usarContar() //resultado “1”

# Closures

É a função que “lembra” de fechar o escopo léxico no qual foi inserido.

function retornaFuncao(){

        const nome = 'Silas'

        return function(){

            return nome

        }

    }

    const funcao = retornaFuncao()

    console.dir(funcao)

No caso desse código, o retorno *nome* é a closure de *retornaFuncao().*

# Funções de callback

São funções executadas ao finalizar uma ação/função. Muito usado em Web quando não sabemos o tempo de execução de uma ação, mas sabemos o que fazer ao finalizar essa ação.

function f1(*callback*){

        console.log('f1')

        if (*callback*) callback()

    }

    function f2(*callback*){

        console.log('f2')

        if (*callback*) callback()

    }

    f1(function(){

        f2(function(){

            console.log('f3')

        })

    })

Dessa forma, garantimos que a função *f2* será executada ao finalizar a função *f1* e, o console.log *f3* será executado ao finalizar *f2*.

# Funções imediatas (IIFE)

São funções que serão executadas imediatamente ao serem declaradas.

(function(){

        console.log('Funções imediatas (IIFE)')

        console.log('execução imediata!')

    })()

# Funções fábrica (Factory Functions)

São funções capazes de criar objetos, através do *return*

function criarPessoa(*nome*, *altura*, *peso*){

        return{

            nome : *nome*,

            altura : *altura*,

            peso : *peso*,

            imc(){

                const indice = *this*.peso / (*this*.altura \*\* 2)

                return indice.toFixed(1)

            }

        }

    }

    const p1 = criarPessoa('Silas', '1.74', '53')

    console.log(`${p1.nome} está com IMC de ${p1.imc()}`)