HISTÓRIA E CONCEITOS

**Currying:** Em ciência da computação, currying é uma técnica de transformação de uma função que recebe múltiplos parâmetros de forma que ela pode ser chamada como uma cadeia de funções que recebem somente um parâmetro cada. Foi inventada por Moses Schönfinkel e Gottlob Frege, e independentemente por Haskell Curry.

/\* ################# CURRYING ################# \*/

function soma(a, b){

    return a + b;

}

soma(100, 3);

soma(100, 1);

soma(100, 7);

// Aplicação //

function soma(a){

    return function(b){

        return a + b;

    }

}

const armazenadaSomaA = soma(100);

armazenadaSomaA(3);

armazenadaSomaA(1);

armazenadaSomaA(7);

**Hoisting:** Ocorre quando seu código Javascript é compilado : Basicamente, todas as declarações de variáveis são movidas para o topo de seu escopo local (se foram declaradas dentro de uma função, por exemplo) ou para o topo do escopo global (se foram declaradas fora de uma função).

Podem ser separadas em dois tipos: variáveis e funções. O *hoisting* de variável só eleva a criação da variável, e não a sua atribuição. Já a função, ela é elevada ao topo, até sua assinatura.

/\* ################# HOISTING ################# \*/

// Aplicação em variável //

function fn(){

    console.log(texto);

    var texto = 'Exemplo';

    console.log(texto);

}

fn();

/\*

O que acontece na verdade com aplicação de Hoisting

quando o javascript é compilado

 function fn(){

    var texto;

    console.log(texto);

    texto = 'Exemplo';

    console.log(texto);

}

fn();

\*/

// Aplicação em função //

function fn(){

    console.log('Hoisting de função');

    function log(value){

        console.log(value);

    }

}

fn();

/\* O que acontece na verdade com aplicação de Hoisting

quando o javascript é compilado

 function fn(){

    function log(value){

        console.log(value);

    }

    console.log('Hoisting de função');

}

fn();

\*/

**Imutabilidade:** Seu conceito mais simples é algo que não pode ser alterado. No conceito de programação a imutabilidade se aplica normalmente a objetos/variáveis que não podem ter seu estado modificado após serem criados, mas isso não garante que os seus valores serão sempre os mesmos. Criar uma variável imutável permite que a referência ao valor na memória seja direta e não exija muito processamento para ser consultado.

/\* ################# IMUTABILIDADE ################# \*/

// EXEMPLO 1 //

const user = {

    name: 'Jacques',

    lastName: 'Cheron'

};

function getUserWithFullName(user){

    return{

        // ECMA Script 2018 - Spread Operator

        ...user,

        //fullName: `${user.name} ${user.lastName}`

        fullName: user.name + ' ' + user.lastName

    }

}

const userWithFullName = getUserWithFullName(user);

console.log(userWithFullName);

// EXEMPLO 2 //

const students = [

    { name: 'Jacques Cheron', grade: 8 },

    { name: 'Luiz Eduardo', grade: 5 },

    { name: 'Wanderley Marques', grade: 10 },

    { name: 'Iza Nakamura', grade: 3}

];

function getApprovedStudents(studentsList){

    return studentsList.filter(student => student.grade >= 7);

}

console.log('Alunos Aprovados:');

console.log(getApprovedStudents(students));

console.log('\nLista de Alunos:');

console.log(students);