**JavaScript**

O que vamos ver no curso?

* Tipos de dados
* Variáveis
* Funções
* Condicionais
* Estruturas de repetição

Data types

* Primitive / Primitive value
* Structural
* Structural Primitiva

**PRIMITIVOS**

* String
* Number
* Boolean
* Undefined
* Symbol
* BigInt

**ESTRUTURAIS**

* Object
  + Array
  + Map
  + Set
  + Date
  + ...
* Function

PRIMITIVO ESTRUTURAL / STRUCTURAL ROOT PRIMITIVE

* null

**SCOPE – ESCOPO**

O scope determina a visibilidade de alguma variável no JS.

//var é global e local, poderá funcionar fora de um escopo de bloco

//hoisting - elevação

 console.log('> existe X antes do bloco?', x)

{

    var x = 0

}

console.log('> existe X depois do bloco?', x)

//const e let são locais e só funcionam no escopo onde foram criadas

let y = 1;

{

    let y = 0

    console.log('> existe Y!', y)

}

console.log('> existe Y depois do bloco?', y)

//const e let são locais e só funcionam no escopo onde foram criadas

const y = 1;

{

    const y = 0

    console.log('> existe Y!', y)

}

console.log('> existe Y depois do bloco?', y)

//No caso da const não dá para declarar um valor isolado, igual o que fizemos no let, é necessário uma declaração em cada scope para que o JS entenda

/\*

## Para criar nomes das variáveis

\*JS é case-sensitive (sensível a maísculas/minúsculas)

\*JS aceita a cadeia de caracteres Unicode

- Posso:

    \* Iniciar com esses caracteres especiais: $ \_

    \* Iniciar com letras

    \* Colocar acentos

    \* Letras maiúsculas e minúsculas fazem diferença

- Não posso:

    \* Iniciar com números

    \* Colocar espaços vazios no nome

- Ideal:

    \* Criar nomes que fazem sentido

    \* Nomes que expliquem o que a variável é ou faz

    \* camelCase

    \* snake\_case

    \* Escrever em Inglês

\*/

//Object

const person = {

    name: 'John',

    age: 30,

    weight: 80,

    isAdmin: true

}

console.log(`${person.name} tem ${person.age} anos.`)

//Array

const animals = [

    'Lion',

    'Monkey',

    'Cat'

]

console.log(animals.length)

**FUNCTIONS – FUNÇÕES**

//FUNCTIONS

//Declaration - declaração da função

//Function statement

function creatPhrase () {

    console.log('Estudar é muito bom!');

    console.log('Paciência e persistência!');

    console.log('Revisão é mãe do aprendizado');

}

//execute, run, call, invoke

creatPhrase()

console.log('Fim do Programa!')

//Função é um liquidificador

function fazerSuco(fruta1, fruta2) {

    return 'Suco de ' + fruta1 + '-' + fruta2

}

const copo = fazerSuco('banana', 'maçã')

console.log(copo);

ARROW FUNCTION

//Arrow Function

const sayMyName = () => {

    console.log('Ricardo')

}

sayMyName()

CALLBACK FUNCTION

//Callback Function

function sayMyName(name){

    console.log('Antes de executar a função callback!')

    name()

    console.log('Depois de executar a callback!')

}

sayMyName(

    () => {

        console.log('Estou em uma callback!')

    }

)

FUNCTION CONSTRUCTOR

Function() Constructor

    \* Expressão new

    \* Criar um novo objeto

    \* This keyword

\*/

function Person(name) {

    this.name = name

    this.walk = function() {

        return this.name + ' está andando!'

    }

}

const rick = new Person('Rick')

const ro = new Person('Rosangela')

console.log(rick.walk())

console.log(ro.walk() + ' E pensando!')

**PROTOTYPE - JS**