**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**

/\*

Prototype

\* prototype-based language

\* prototype chain

\* \_\_proto\_\_

Type conversion (typecasting) vs Type coersion

\* Alteração de um tipo de dado para outro tipo.

\*/

// Manipulando Strings e Números

// Transformar String em Números e Números em String

let string = '123'

Number(string)

console.log(string)

let number = 321

String(number)

console.log(number)

// Contar quantos caracteres tem uma palavra e quanto dígitos tem um número

let word = 'Anticonstitucionalicímamente'

console.log(word.length)

let numero = 12345

console.log(String(numero).length)

// Transformar um número quebrado com 2 casas decimais e trocar ponto por vírgula

let numeral = 345.7080110558

console.log(numeral.toFixed(2).replace('.', ','))

// Transforme letras minúsculas em maiúsculas. Faça o contrário disso também!

let bigWord = 'Programar é muito bacana!'

console.log(bigWord.toUpperCase())

// Separe um texto que contem espaços, em um novo array onde cada texto é uma posição do array. Depois disso, transforme o array em um texto e onde eram espaços, coloque \_

let phrase = 'Eu quero viver o Amor!'

let myArray = phrase.split(' ')

let phraseWithUnderScore = myArray.join('\_')

console.log(phraseWithUnderScore.toLocaleUpperCase())

// Verificar se o texto contém a palavra Amor

let frase = 'Eu quero viver o amor!'

console.log(frase.includes('amor'))

// Criar Array com construtor

let myNewArray = new Array('R', 'i', 'c', 'a', 'r', 'd', 'o')

console.log(myNewArray)

// Contar elementos de uma Array

console.log(['a', 'xoxota', 'mega\_sena\_da\_virada'].length)

// Transformar uma cadeia de caracteres em elementos de um Array

let toWord = 'manipulação'

console.log(Array.from(toWord))

// Manipulando Arrays

let techs = ['html', 'css', 'js']

// adicionar um item no fim

techs.push('nodejs')

// adicionar no começo

techs.unshift('sql')

// remover do fim

techs.pop()

//remover do começo

techs.shift()

// pegar somente alguns elementos do array

//console.log(techs.slice(1))

// remover 1 ou mais itens em qualquer posição do array

techs.splice(2)

// encontrar a posição de um elemento no array

let index = techs.indexOf('css')

techs.splice(index, 1)

console.log(techs)

**EXPRESSÕES E OPERADORES**

Expressions and Operators

- Binary

- Unary

- Ternary

/\*

new

is a left-hand-side expression

to create a new object

\*/

let name = new String('Rick')

let age = new Number(49)

console.log(name, age)

let date = new Date('2022-12-29')

console.log(date)

/\* Unary Operators

    typeof

    delete

\*/

const person = {

    name: 'Rick',

    age: 49,

}

delete person.age

console.log(person)

//Operadores Aritiméticos

    //multiplicação \*

    console.log(3.2 \* 5.5)

    //divisão /

    console.log(12 / 2)

    //soma +

    console.log(34 + 67)

    //subtração -

    console.log(34 - 23)

    //resto da divisão %

    let remainder

    remainder = 11 % 11

    console.log(remainder)

    //incremento ++

    let increment = 0

    increment ++

    console.log(increment)

    //decremento --

    let decrement = 2

    decrement --

    console.log(decrement)

    //exponencial \*\*

    console.log(3 \*\* 3)

**Grouping Operator**

// Grouping Operator ()

let total = (2 + 3) \* 5;

console.log(total)

**Operadores de Comparação**

// Operadores de Comparação

// Irá comparar valores e retornar um boolean como resposta à comparação

let one = 1

let two = 2

// ==  igual a

console.log(  two == 1)

console.log( one == "1")

// !=  diferente de

console.log(one != two)

console.log(one != 1)

console.log(one != "1")

// Operadores de Comparação

// Irá comparar valores e retornar um boolean como resposta à comparação

let one = 1

let two = 2

// ==  igual a

console.log(  two == 1) **FALSE**

console.log( one == "1") **TRUE**

// !=  diferente de

console.log(one != two) **TRUE**

console.log(one != 1) **FALSE**

console.log(one != "1") **FALSE**

//  ===   estritamente igual a

console.log( one === "1") **FALSE**

console.log( one === 1) **TRUE**

//  === estritamente diferente de

console.log( two !== "2") **TRUE**

console.log( two !== 2 ) **FALSE**

// >  Maior que

console.log(one > two) **//FALSE**

// >= Maior ou igual a

console.log(one >= 1) **//TRUE**

console.log(two >= 1) **//TRUE**

// <  Menor que

console.log(one < two) **//TRUE**

// <= Menor ou igual a

console.log(one <= two) **//TRUE**

console.log(one <= 1) **//TRUE**

console.log(one <= 0) **//FALSE**

**// Operadores de atribuição (Assignment)**

let x

console.log(x)

// assignment

x = 1

console.log(x)

// addition assignment

x += 2

console.log(x)

// subtraction assignment

x -= 1

console.log(x)

// multiplication assignment

x \*= 2

console.log(x)

// division assignment

x /= 2

console.log(x)

// remainder, exponetiation

x %= 2

console.log(x)

**OPERADORES LÓGICOS**

// Logical Operators

// 2 valores booleanos, quando verificados, resultará em verdadeiro ou falso

let pão = true

let queijo = false

// And &&

console.log(pão && queijo)

// Or ||

console.log(pão || queijo)

// Not !

console.log(!pão)

**Operador Condicional Ternário**

// Operador Condicional (Ternário)

// Dependendo da condição, nós receberemos valores diferentes

// Condição então valor 1 se não valor 2

// condition ? value1 : value2

// Exemplos

// Café da manhã top

let pane = true

let queso = true

const niceBreakfast = pane && queso ? "café top!" : "café ruim pra caramba!"

console.log(niceBreakfast)

// Maior de 18

let age = 15

const canDrive = age >= 18 ? "You can drive!" : "You cannot drive!"

console.log(canDrive)

/\*

    FALSY

    Quando um valor é considerado false em contextos onde um booleano é obrigatório (condicionais e loops)

    false

    0

    -0

    ""

    null

    undefined

    NaN

\*/

/\*

    TRUTHY

    Quando um valor é considerado true em contextos onde um booleano é obrigatório (condicionais e loops)

    true

    {}

    1

    3.23

    "0"

    "false"

    -1

    Infinity

    -Infinity

\*/

**OPERATOR PRECEDENCE**

/\*

    Operator Precedence

    \* Grouping                        ( )

    \* Negação e incremento            ! ++ --

    \* Multiplicação e divisão         \* /

    \* Adição e subtração              + -

    \* Relacional                      < <= > >=C

    \* Igualdade                       == != === !==

    \* AND                             &&

    \* OR                              ||

    \* Condicional                     ?:

    \* Assignment (atribuição)         = += -= \*=

\*/