JavaScript

A importância da sintaxe
* Toda linguagem tem
* Uma boa comunicação necessita de uma boa sintaxe
* 82% dos erros para iniciantes de programação

Maneiras de usar o JavaScript
* Console
* No próprio HTML ex:
<script>
console.log('Olá mundo');
<script>
* Criar pasta e inserir no HTML e no final do Body colocar <script src=" #."><script>

Criando comentários no JS
// comentário em linha
/* */ comentário multi linhas

Tipos de dados
 Gramática do js
*Saber os elementos da linguagem e suas combinações
*A arte de escrever corretamente
Vocabulário
*Conjunto de termos e expressões
* Agrupamento de palavras

String (Cadeia de caracteres) ex: a, b, c ...
" " : Aspas dupla
' ' : Aspas simples
` ` : Template literals ou Template strings
Number (números)
33 * inteiros
12.5 * reais
Nan * not a number
Infinity * infinito
Boolean
* Somente 2 valores
true *verdadeiro
false *falso
undefined
*Indefinido
null
*nulo
*Objeto que não possui nada dentro
* diferente de indefinido
Object (dado que nós da estrutura)
* Objeto
*Propriedades/Atributos
*Funcionalidades / Métodos

{propriedade: "valor"}
Array
* uma lista
*Agrupamento de dado
["leia", 35]

Tipos de dados
Conforme o ECMAScript standard temos 9 tipos de dados:

Data types
*Primitive / Primitive value
*Structural
*Structural Primitive
Primitivos
*String "ABC" "BCA" "bac" "bca"...
*Number (10, 10.2,)
*Boolean
*Undefined
*Symbol
*BigInt
Estruturais
*Object
*Map
*Set
*Date
*Function
Primitivo Estrutural / Structural Root Primitive
*null

Variáveis
Nomes simbólicos para receber algum valor
Atalhos de código
Identificadores
3 palavras reservadas para criar uma variável

- **var**
- **let**
- **const**

O js é uma linguagem fracamente tipada e dinâmica.
* Variáveis não precisam ter um tipo previamente definido
* Podemos mudar o conteúdo da variável

obs: Para ver o tipo de variável usa-se o typeof

Scope
* Escopo determina a visibilidade de alguma variável no js
Block statement
(declaração de bloco)
// vamos iniciar um bloco
{
 aqui dentro é um bloco e posso colocar qualquer código
} aqui fechamos o bloco
O bloco também criará um novo escopo. Chamamos de `block-scoped`

Criar nomes de Variáveis
JS é case-sensitive (sensível ao caso)
JS aceita a cadeia de caracteres Unicode
*** Posso:**
- Iniciar com esses caracteres especiais: $ _
- Iniciar com letras
- Colocar acentos
- Letras maiúsculas e minúsculas e fazem diferença

*** Não posso:**
- Iniciar com números
- Colocar espaços vazios no nome

*** Ideal**
- Criar nomes que fazem sentido
- Que explique o que a variável é ou faz
- camelCase
- snake_case
- Escrever em inglês
-

Ex: se você for falar de graus , use o nome celsius

```
// variáveis e tipos de dados
// declaração or declaration de var
var name

// atribuir valores or assignment

name ="leia";

// que tipo de dado foi colocado na variável

console.log(typeof name)

// agrupamento de declarações
let age =20
let isHuman =true

let age, isHuman
```

```
age = 18
isHuman = true

console.log(name, age, isHuman);// multiplos argumentos

let name = "leia"

let age, isHuman

age = 18
isHuman = true

console.log(name, age, isHuman);// multiplos argumentos

// concatenando valores
console.log(' A ' + name + ' têm ' + age + ' anos.');

// interpolando valores com template literals or template strings
console.log(`A ${name} têm ${age} anos.`);

// object

const person = {
    name: 'Léia',
    age: 35,
    weight: 84.4,
    isAdmin : true
}
console.log(person.name)
console.log(person)

// Array
const animals = [
'lion',
'monkey',
'cat'
]
// acessar valores
console.log(animals)
console.log(animals[0])
console.log(animals.length)
```

//EXERCÍCIOS

// 1. Declare uma variável de nome weight

```
//let weight;
```

// 2. Que tipo de dado é a variável acima?

```
//console.log(typeof weight);

/* 3. Declare uma variável e atribua valores para cada um dos dados:
 *name : String
 * age : Number (integer)
 * stars : number (float)float numero quebrado
 * isSubscribed: Boolean

*/

/*let name = 'Leia';
let age = 35;
let stars = 5.35;
let isSubscribed= true;*/

/* 4. A variável student abaixo é de que tipo de dados?
Resposta:
let student ={}; // object
console.log( typeof student)

4.1 Atribua a ela as mesmas propriedades e valores do exercício 3.
Resposta:
let name = 'Leia';
let age = 35;
let weight = 5.35;
let isSubscrived= true;
let student ={
   name : 'Leia',
   age : 35,
   weight :5.35,
   isSubscribed: true
}
console.log(student);

4.2 Mostre no console a seguinte mensagem:
<name> de idade <age> pesa <weight> Kg.

Atenção, substitua <name> <age> e <weight> pelos valores de cada propriedade do objeto
Resposta:
let student ={
   name : 'Leia',
   age : 35,
   weight:84.4,
   isSubscribed: true
}
console.log(`${student.name} de idade ${student.age} pesa ${student.weight} Kg`);

5. Declare uma variável do tipo array, de nome students e atribua a ela nenhum valor, ou seja, somente o Array vazio.
Resposta:
```

```
let students = []
console.log(students)
```

6. Atribua valor para a variável acima, colocando dentro dela o objeto student da questão 4. (Não copiar e colar o objeto, mas usar o objeto criado e inserir ele no array)

```
let student ={
   name : 'Leia',
   age : 35,
   weight:84.4,
   isSubscribed: true
}

let students = []


students = [
   student
]

console.log(students);
```

7. Coloque no console o valor da posição zero do array acima
Resposta:

```
let student ={
   name : 'Leia',
   age : 35,
   weight:84.4,
   isSubscribed: true
}

let students = []


students = [
   student
]

console.log(students);
console.log(students[0])
```

8. Crie um novo student e coloque na posição 1 do array students
Resposta:

```
let student ={
   name : 'Leia',
   age : 35,
   weight:84.4,
   isSubscribed: true
}
```

```
let students = []


students = [
  student
]

console.log(students);
console.log(students[0])
const leia ={
    name: "leia",
    age: 35,
    weight: 84,
    isSubscribed: true,

}

students = [

  student,
  leia,

]

console.log(students);
```

9. Sem rodar o código responda qual é a resposta do código abaixo e por que após sua resposta, rode o código e veja se você acertou.
Resposta: vai dar undefined, o a ainda não foi declarado

```
console.log(a)
var a =1
```

Functions

```
//Funções de uma function, agrupamento de código e reutilização de código, dar significado a um bloco de código
// criar um aplicativo de frases motivacionais
// declarar a função

function createPhrases(){

 console.log('Estudar é muito bom');
 console.log('Paciência e persistência');
 console.log('Revisão é mãe do aprendizado');

}

createPhrases()

// executando a função
```

```
console.log('Fim do Programa');


/*
//você pode declarar funções dentro de variáveis

//function anonymous

//parâmetros

const sum = function( number1, number2){
    console.log(number1+number2)

}

sum(2, 3) //argumentos
sum(20, 5)*/


const sum = function( number1, number2){
    total = number1+number2
    return total

}

let number1 = 34
let number2 = 25

console.log(`o número 1 é ${number1}`)
console.log(`o número 2 é ${number2}`)
console.log(`a soma é ${sum(number1, number2)}`)

// Jeito ludico de entender o que é função.Função é um liquidificador

function fazerSuco(fruta1, fruta2) {
   return fruta1 + fruta2
}

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

console.log(copo);

// function scope

let subject ='create video'

function createThink() {
    subject = 'study'
   return subject
}
```

```
console.log(createThink(subject))
console.log(subject)

// function hoisting

sayMyName()
function sayMyName() {
    console.log('leia')
}
sayMyName()

// arrow function

const sayMyName =() =>{
    console.log('leia')
}
sayMyName()

// 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
*expressão new
*criar um novo objeto
*this keyword

 function Person(name) {
    this.name = name
 }
 const leia = new Person("leia")
 const clara = new Person("Clara")
 console.log(leia)
 console.log(clara)

Prototype
*prototype-based laguage
*prototype chain
*_proto_
```

Type conversion (typecasting) vs Type conversion
*Alteração de um tipo de dado para outro tipo

Manipulando Strings e Números
Transformar string em número e número em string
```
let string='123'
console.log(Number(string))
let number = 321
console.log(String(number))
```

Contar quantos caracteres tem uma palavra e quantos dígitos tem um número
```
let word = "Paralelepipedo";
console.log(word.length);
let number = 123456789
console.log(String(number).length)
```

Transformar um numero quebrado com 2 casas decimais e trocar ponto por vírgula
```
let number = 345.33452345
console.log(number.toFixed(2).replace(".",","))
```

Transforme letras minúsculas em maiúsculas. Faça o contrário disso também
```
let word = "Programar é muito bacana!";
console.log(word.toUpperCase());

let minúsculas = "Programar é muito bacana";
console.log(minusculas.toLowerCase())
```

Verificar se o texto contém uma palavra específica
```
let phrase = "É preciso saber viver";
console.log(phrase.includes("amor"))

let phrase1 = "É preciso saber viver";
console.log(phrase1.includes("viver"))
```

Separe um texto que contém 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 texto = "A vida é bela";

let myArray = texto.split(" ");

let textoWithUnderscore = myArray.join("_")
console.log(textoWithUnderscore)
```

Manipulando Arrays
Criar Array com construtor
```
let myArray = ['a','b','c'];
console.log(myArray);
```

Contar elementos de um array

```
console.log([
    "a",
    {type:"array"},
    function () {
            return "ola"
    }
    ].length)
```

Transformar uma cadeia de caracteres em elementos de um array

```
let word = "manipulação"
console.log(Array.from(word))

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, 3))
```

remover 1 ou mais itens em qualquer posição do array

```
techs.splice(1, 1)
```

encontrar a posição de um elemento array

```
let index =techs.indexOf('js')
console.log(techs)
console.log(index)
```

Expressões e Operadores

-Expressions

-Operators

- Binary
- Unary
- Ternary

```
let number = 1// exemplo de expressões
console.log(number + 1)
```

new

* left-hand-side expression
*criar um novo objeto

```
let name = new String('Leia')
name.surName = "Santos"
let age = new Number(30)
console.log(name, age)

let date = new Date('2022-05-29')
console.log(date.__proto__)
```

Operadores unários
- typeof
- delete

typeof
```
console.log(typeof "leia")
console.log(typeof 12)
console.log(typeof true)
console.log(typeof'10')
```

delete
```
const person = {
    name: 'Leia',
    age: 30,
    peso: 84,
}
delete person.peso
console.log(person)
```

Operadores Aritméticos
multiplicação *
```
console.log(4 * 2)
console.log(4.2 * 2)
console.log(4.2 * 2.5)
```

divisão /
```
console.log(10 / 5)
console.log(250 / 5)
console.log(10 / 2)
```

soma +
```
console.log(10 + 10)
```

subtração -
```
console.log(50 - 25)
```

resto da divisão %
```
let remainder
remainder = 11 % 3
console.log(remainder)
```

incremento ++ (significa adicionar mais um número)

```
let increment = 0
increment++
console.log(increment)
```

decremento --

```
let decrement = 0
decrement--
console.log(decrement)
```

exponencial **

```
console.log(2 ** 2)
console.log(3 ** 10)
```

Grouping operator ( )

```
let total = (2 + 3) * 5
console.log(total)
```

Operadores de comparação

Irá comparar valores e retornar um Boolean como resposta a 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")
```

=== estritamente igual a

```
console.log( one === "1")
console.log( one === 1)
```

!== estritamente diferente de

```
console.log( two !==  "2")
console.log( two !== 2)
```

> Maior que

```
console.log(one > two)
```

>= Maior igual a

```
console.log(one >= 1)
console.log(two >= 1)
```

< Menor que

```
console.log(one < two)
```

<= Menor igual a

```
console.log(one <= two)
console.log(one <= 1)
```

```
console.log(one <= 0)
```

Operadores de atribuição (Assignmet)

```
let x
assignment
x = 1
console.log(x)
addition assignment
x = 1
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, resulta em verdadeiro ou falso

```
let pao = true
let queijo =true
AND &&
console.log(pao && queijo)

 OR ||
let pao = false
let queijo =true
ou
let pao = true
let queijo =true
console.log(pao || queijo)

NOT !
let pao = false
let queijo =true
console.log(!pao)
```

Condição então valor 1 se não valor 2

```
condition ? value: value2
```

Exemplos

Café da manhã top

```
let pao = true
let queijo = true
```

```
const niceBreakfast = pao && queijo ? 'Café top' : ' Café ruim'
console.log(niceBreakfast)
let pao = false
let queijo = true
const niceBreakfast = pao || queijo ? 'Café top' : ' Café ruim'
console.log(niceBreakfast)

Maior que 18
let age = 16
const canDrive = age >= 18 ? 'can drive ' : 'cannot drive'
console.log(canDrive)

 Operador de String (String operator)
comparison ( comparação)
console.log('a' == 'a')

 concatenation (concatenação)
Retorna a união de duas Strings
console.log('a' + 'a')
let alpha = 'alpha'
console.log(alpha + 'bet' )

Type conversion (typecasting) vs Type coersion
console.log(Number('9') + 5)

Falsy
Quando um valor é considerado false em contextos onde um booleano é obrigatório( condicionais e
loops)
false
0
-0
""
null
undefined
NaN
console.log(true ? 'verdadeiro' : 'falso')
console.log("" ? 'verdadeiro' : 'falso')
console.log(null ? 'verdadeiro' : 'falso')
console.log(undefined ? 'verdadeiro' : 'falso')

TRUTHY
Quando um valor é considerado true em contextos onde uma booleano é obrigatorio( condicionais e
loops)
true
{}
[]
1
3.23
"0"
"false"
```

-1
Infinity
-Infinity
console.log({} ? 'verdadeiro' : 'falso')
console.log([] ? 'verdadeiro' : 'falso')
console.log(3.23 ? 'verdadeiro' : 'falso')

Operator precedence (Precedência de operadores)
* grouping ()   ex: console.log((2+5) * 10)
*negação e incremento ! ++ --
*multiplicação e divisão * /
*adição e subtração + -
*relacional < <= > >=c
*igualdade == != === !==
*AND
*OR
*condicional ||
*assignment(atribuição)

if ... else
Exemplos:
```
let temperature = 36.5
if(temperature >= 37){
   console.log('Febre')
}else{
   console.log('Saúdavel')
}


let temperature = 37.5
if(temperature >= 37.5){
   console.log('Febre alta')

}else if(temperature < 37.5 && temperature >= 37){
 console.log('Febre moderada')

} else{
    console.log('Saúdavel')
}
```

Switch
Exemplos:
```
let expression = 'a'
switch (expression){
    case 'a':
        // código
        console.log('a')
    break
case 'b':
   // código para expression b
```

```
   console.log('b')
   break
 default:
    console.log('default')
  break
}

function calculate(number1, operator, number2) {
    let result

    switch (operator){
            case '+':
               result = number1 + number2
              break
            case '-':
               result = number1 + number2
              break
            case '*':
                result = number1 + number2
              break
            case '/':
               result = number1 + number2
              break
       default:
         console.log('não implementado')
           break

    }

    return result
}

console.log(calculate(4, '+', 8))

throw
```

Estrutura de repetição
for

- break - para a execução do loop
- continue - pula a execução do momento

Exemplo:

```
for(let i = 10; i > 0; i--){
    if(i === 5){
            continue;
    }
    console.log(i)
}
```

while(enquanto)
Usa quando não sabemos o momento exato da parada.
Exemplo:

```
let i = 0;
while(i < 10){
    console.log(i)
    i++;
}

for ...of
let name ='Leia'
let names = ['jessica', 'manuela', 'natália']
for(let name of names){
   console.log(name)
}

for...in
let person ={
    name: 'Clara',
    age: 24,
    weight: 80
}
for(let property in person){
    console.log(property)
    console.log(person[property])
}
```