JAVASCRIPT: TIPOS, VARIAVEIS E FUNÇÕES.

Video 01 - type_Number.js

```
const meuNumero = 3;

const primeiroNumero = 1;
const segundoNumero = 2;

const operacaoMatematica = primeiroNumero + segundoNumero;

console.log(operacaoMatematica)
```

PONTO FLUTUANTE

```
const numeroPontoFlutuante = 3.3;
const pontoFlutuanteSemZero = .5

const novaOperação = primeiroNumero/numeroPontoFlutuante

console.log(novaOperação)
```

Math.round() - faz o arredondamento de um numero decimal.
Math.PI()
Math.ceil() que retorna o maior número inteiro que é maior que o número passado, por exemplo Math.ceil(11.123), o valor fica 12
Math.floor() que retorna o menor número inteiro que é menor que o número passado, por exemplo Math.floor(11.789), o valor fica 11

Formatando o resultado com ajuda do método .toFixed()
É possivel controlar o número de casas decimais após a vírgula, de forma mais simples, indicando o número de casas que eu quero como parâmetro.

Um ponto importante de se observar é que o método .toFixed() arredonda o número para cima
Outro ponto importante de se observar é que seu retorno será uma string representando o número.

USO DO METODO .toFixed()

```
function ganhoPorHora(salario, horasTrabalhadasNoMes) {

const salarioHora = (salario / horasTrabalhadasNoMes);

const total = salarioHora.toFixed(2);

return total;
};

verificaGanho = ganhoPorHora(555,6)
console.log(verificaGanho)
```

-FORMATANDO O RESULTADO PARA REAL.

No JavaScript temos um método chamado toLocaleString() que converte um número para uma string, já tratando a questão do arredondamento e convertendo para a moeda do país que queremos, no nosso caso, o real, tornando a tarefa do programador muito mais simples.

O método toLocaleString() recebe alguns argumento - um objeto literal com as propriedades -, no meu caso eu utilizei:

style : Que é o estilo do formato a ser utilizado, aqui é permitido usar:

decimal para representar números simples.

currency que diz respeito ao formato monetário e que vai indicar a moeda que vai ser utilizada.

percent para formato percentual.

currency: A moeda para usar na formatação monetária

```
function ganhoPorHora(salario, horasTrabalhadasNoMes) {

const salarioHora = (salario / horasTrabalhadasNoMes);

const formatado = salarioHora.toLocaleString('pt-BR', { style:
'currency', currency: 'BRL' });

return formatado;

}
let verificaReal = ganhoPorHora(1000,8);
console.log(verificaReal)
```

```
NaN -> Not A Number (não é um número)

Erro será apresentado se houver operações matematica entre string com number.
```

ORDENAÇÃO DE NUMEROS COM .sort()

```
var lista = [10,1, 5, 9, 8, 12, 15];
OBS: função opcional
function comparaNumeros(a,b) { if (a == b) return 0; if (a < b) return -1; if (a > b) return 1; }
```

```
O sort recebe opcionalmente uma função de comparação que, dados dois valores, deve devolver um número inteiro:

Se for 0 indica que são iguais
Se for -1 indica que o primeiro valor é menor
Se for 1, o segundo é menor.
```

```
Simplicando a implementação Com arrow functions
let ordenaNum = lista.sort((a, b) => a - b);
console.log(ordenaNum)
```

TIPO STRING

```
Metodo - toLowerCase()
Converte todas letras em minúsculas

Propriedade .length
verifica a quantidade de caracteres uma string possue.
converter qualquer coisa em String
String(coisa)
```

TIPO BOOLEAN

```
const primeirNumero = 5;
const segundNumero = 10;
// dois os tres iguai é uma comparação.
//console.log(primeirNumero === segundoNumero)

const texto1 ="alura";
```

```
const texto2 ="Alura";

console.log(texto1===texto2)
```

Git

criamos um novo repositorio com git init
depois git status
proximo passso- git remote add origin ..... cola o endereço do git...
proximo os tres ultimos comandos -> git add ./ git commit -m " exemplos da aula 01"/ git push origin master

VIDEO 01 ARQUIVO - script.js

var

Var pode ser usada antes de ser declarada, e ser declara posteriormente, porque o JavaScript na inicialização do programa vai carregar as var primeiro.

```
var altura1 = 12;
var comprimento1 = 5;

area1 = altura1 * comprimento1;
console.log(area1)
var area1;
```

let

para o let funcionar dentro do bloco de codigo a variavel dever ser inicializada no começo do programa, ou antes de utiliza-las em um bloco de codigo, assim a variavel vai fazer as operações dentro do bloco e vai armazenar o valor na let que esta fora do bloco de codigo, podemos alterar o valor dela caso necessário.

```
let forma2 = 'retângulo'
let altura2 = 5;
let comprimento2 = 7;
let area2;

if (forma2 ==='retângulo'){
area2 = altura2 * comprimento2;
} else{
area2 = (altura2 * comprimento2)/2;
}

console.log(area2)
```

const

constante porque não podemos mudar o valor dela.

Não podemos utilizar esse tipo de variável em algo que vai precisar mudar de valor, neste caso só será viável em casos de valores fixos.

```
const forma3 = 'quadrado';
    const altura3 = 5;
  const comprimento3 = 8;
        let area3;

 if (forma3 === 'quadrado'){
area3 = altura3 * comprimento3
            } else{
area3 = (altura * comprimento3)/2;
                }

console.log(area3)
```

Video 03 - truthy e falsy/ arquivo truthy-falsy.js

boolean

```
const usuarioLogado =true
 const contaPaga = false;
```

truthy ou falsy

o javascript condidera esses resultados como tipo falso.
0 - falsy - zero
"" - falsy - string vazia
null - ele reprenta vazio
undefined - variavel sem valor

```
console.log ( 0 == false)
console.log ( '' == false)
 console.log ( 1 == true)
```

null => vazio ou nada - Exemplos logo abaixo

```
let minhaVar;
let varNull = null;
```

```
let numero2 = 3;
let texto = 'Alura'


console.log(typeof numero2)
console.log(typeof texto)
```

metodo typeof => serve para verificar qual o tipo de dado esta sendo guardada na variável em questão.

Vídeo 04 conversões de tipos/ arquivo conversoes.js

booleanos

CONVERSÃO IMPLICITA

```
const numero = 322;
const numeroString = "322" // é possivel também colocar a função Number(numeroString) na variavel.
/*
= - 1 igual é para atribuição
== - comparação simples dos valores
=== - compara os valores e tipo de dado entre as comparações.
```

CONVERSÃO EXPLICITA

```
Number() - tranforma uma string em numero
String() -  transforma numero em string
*/
console.log(numero + Number(numeroString))
```

Aula 03 - video 02/ arquivo script02.js

TIPAGEM DINÂMICA

o Javascript aceita a troca do tipo de dado de uma variável, e untyped, significa que não precisamos declarar o tipo dessa variável, como number, string ou booleano e etc.

EXEMPLOS:

```
let minhaVAr = 876;
minhaVAr = "texto";
minhaVar = true;
```

MULTIPARADIGMA:

utilizado para resolver vários problemas do nosso cotidiano.

POR QUE SE CHAMA JS?
porque veio do mesmo fundado do Java, nome dado para dar maior credibilidade ao Javascript no início de sua utilização.

O QUE É ES6/ES2015?
É a versão do Javascript de 2015, com as atualizações das funções empregadas na linguagem.

LIGUAGEM INTERPRETADA:
sua execução não precisa passar pelo copilador, ela vai ser executada por outro programa na hora.

O QUE É O NodeJS?
NodeJS é um intrepretador de codigo de JavaScript, vai rodar os programas feitos em JavaScript para tester o nosso codigo utilizando a função console.log()

AULA 03 - VIDEO 03/ ARQUIVO erros.js

```
const numero =1;
```

const numero;
Erro na declaração de variável Const, elas não podem ser criadas sem algum valor, vai aparecer como erro de (SyntaxError: Missing initializer in const declaration) falta de inicializador na variavel const.

console.log(numero
Na falta de alguma infromação vai apresentar o erro: (SyntaxError: missing) after argument list)

TIPOS DE ERROS

RangeError: Quando o código recebe um dado do tipo certo, porém não dentro do formato aceitável. Por exemplo, um processamento que só pode ser feito com números inteiros maiores ou iguais a zero, mas recebe -1.

ReferenceError: Normalmente ocorre quando o código tenta acessar algo que não existe, como uma variável que não foi definida; muitas vezes é causado por erros de digitação ou confusão nos nomes utilizados, mas também pode indicar um erro no programa.

SyntaxError: Na maior parte dos casos ocorre quando há erros no programa e o JavaScript não consegue executá-lo. Os erros podem ser métodos ou propriedades escritos ou utilizados de forma incorreta, por exemplo, operadores ou sinais gráficos com elementos a menos, como esquecer de fechar chaves ou colchetes.

TypeError: Indica que o código esperava receber um dado de um determinado tipo, tal qual uma string de texto, mas recebeu outro, como um número, booleano ou null.

AULA 03 - ARQUINO - console.js

```
const minhaVar = true;
```

log(): é o registro do nosso console, podendo colocar qualquer informações, como string, number, variável.

EX.

```
console.log(245)
console.log("eu sou texto")
console.log(minhaVar2)
```

TRATAMENTO DE ERRO.

```
console.erro("deu erro!")
```

varios tipos de consoles para cada tipo de erro que será exibido para o usuario no momento de utilizar algum recurso de um programa inadequadamente.

Outros métodos, existem:

console.table() para visualizar de forma mais organizada informações tabulares;
console.time() e console.timeEnd() para temporizar período que uma operação de código leva para ser iniciada e concluída;
console.trace() para exibir a stacktrace de todos os pontos (ou seja, os arquivos chamados) por onde o código executado passou durante a execução.

AULA 04 - VIDEO 01 - ARQUIVO comparacoes.js

Comparação implícita:

Javascript vai converter o numero em texto e vai ser verdadeiro no console.

```
const nuemro =5;
const texto = '5';
```

console.log(numero == texto); => Só compara Valores.
console.log(numero === texto); => Compara o valor e o tipo de dado.

As boas praticas pedem para fazer a conversão explícita utilizando:

```
Number()
String()
```

```
*/
const numero =5;
const texto = '5';

console.log("Console comparações - Primeira analise ==:", numero == texto);
console.log("Console comparações - Segunda analise ===:" ,numero === texto);
```

AULA 04 - VIDEO 02 - ARQUIVO - ternario.js

```
const idadeMinima =18;
const idadeCliente =16;

    if classico.
if (idadeCliente >= idadeMinima){
    console.log("Suco de uva")
        }else{
    console.log("refrigerante")
        }
        }
```

OPERADOR TERNARIO

CONDIÇÃO

```
console.log("console operador ternario", idadeCliente >= idadeMinima ?
```

TRUE → `"Suco de uva":` FALSE → `"Refrigerante")`

Por que é um operador ternário?
porque temos três operadores sendo utilizados em uma única função, como inicialmente temos a condição de teste, se a => é maior ou igual a b, e logo após o resultado o ***(?)*** temos dois resultados possíveis, true : ou false.

AULA 04-VIDEO 03/ ARQUIVO template-string.js

Ela surgiu para facilitar no processo de concatenação, é um método novo que veio a partir do ES6.

EXEMPLO DE COM CONCATENAÇÃO

```
const nome = "Bruno"
const idade= 2022-1994
const cidade = "Pititinga"
```

```
const apresentacao1 = "meu nome é " + nome + ', minha idade é '+ idade +
                    " e nasci na cidade de " + cidade;

   console.log('console do template - Primero teste:', apresentacao1)

                      // TEMPLATE STRING - metodo.
const apresentacao2 = `meu nome é ${nome}, minha idade é ${idade} e nasci
                        na cidade de ${cidade}`

   console.log(' console do template - Segundo teste:', apresentacao1)
```

AULA 05 VIDEO 01 - ARQUIVO funcoes.js

- O uso da função é para que parte de um código seja executado apenas quando desejamos que ele execute. - Funções tem dois momentos, primeiro ela é declarada, depois é quando executamos a função.

```
                          let x = ''
                        console.log(x)
                          x = 'oi';
```

Exemplo de funções.

```
                    function imprimeTexto(texto){
              console.log("Exemplo da Aula05-Video01:", texto)
                                    }
                     imprimeTexto('Ola Mundo!')
                    imprimeTexto('outro texto')
                        imprimeTexto(34-20)

                          function soma1(){
                       return resultado = 2 + 2;
  return tem que ser a ultima linha do codigo antes de fechar o bloco.
                                    }
              Podemos utilizar funções dentro de outra funções.
                        imprimeTexto(soma1())
```

ver funções prontas de matemática do JS - https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Math#description

AULA 05 - VIDEO02 ARQUIVO parametros.js

```
function nome(parametro1, parametro2...){
                código....
                 exemplo:
          retrun parametro1 + parametro2

           function soma(num1, num2,){
               return  num1 + num2;
                        }
             console.log(soma(2,2))

          function nomeIdade(nome, idade){
return `Console2 Exemplo Video 02 parametros.js - resultado: meu nome é
            ${nome} e minha idade é ${idade}`;
                        }

           console.log(nomeIdade('Bruno', 27))

         function multiplica(num1 = 1 ,num2 = 1){
                  return num1 * num2;
                        }
          console.log(multiplica(soma(4, 5), ))
```

Não podemos colocar apenas um parâmetro para a função que deve receber dois parâmetros, vai ocorrer NaN, podemos fazer dessa seguinte maneira:

```
function multiplica(num1 = 1, num2 = 1){
          return num1 * num2;
                }
```

Nesta configuração podemos colocar apenas um parâmetro e não vai do erro. Os parâmetros podem receber qualquer nome, mesmo que se repitam, eles só vão ser validos dentro da função, podemos ter várias funções com os mesmos nomes de parâmetros iguais, e também pode ter vários parâmetros.

PARAMETROS X ARGUMENTOS

Parâmetros é o momento em que a função é definida (no caso, num1, num2, etc.) e argumentos como os dados que utilizamos para executar a função;

multiplica (2,10)

saída: 20

AULA 05 VIDEO 03 - ARQUIVO - expressoes-funcoes.js

Expressão de função, a função não tem um nome, ela fica em uma const, chamada também função anônima.
Ela não pode ser chamada antes de ser declarada, porque ela vai se comportar como uma variável const, sendo assim devemos declarar antes para depois fazer uso desse tipo de função, vai da erro caso tentarmos chamar essa função antes dela ser declarada.
obs: Funções e var são "Listadas" no topo do arquivo antes de rodar o programa.

```
const soma3 = function(num1, num2){  return num1+num2}
                console.log(soma3(1,1))
               console.log(aparesentar())
                function aparesentar(){
                     return "olá"
                           }
```

AULA05 - VIDEO 04 ARQUIVO arrow-function.js

Essa função utiliza uma seta como característica desse método, ela não vem nomeada e sempre é necessário ser declarada com uma const, não precisa de retorno quando for escrita com apenas uma linha de código, se precisar mais de uma linha vai precisar de return.

Arrow function

```
const aparesentarArrow = nome => `meu nome é ${nome}`;

      const soma = (num1, num2) => num1 + num2;

    //Arrow function com +  de uma linha de instrução

      const somaNumerosPequenos = (num1, num2) => {
               if(num1 > 10 || num2 > 10){
            return "somante numeros de 1 a 9"
                         }else{
                   return num1 + num2;
                         }
                         }
```

Não confundir com o operador maior ou igual que - (>=)
Como saber quando utilizar cada um dos tipos de funções?(Função, Expressão de função, e Arrow Function)
Arrow Function é um jeito mais rapido e mais curto de criar uma função, deixando com poucas linhas de codigos, tem grandes vantagens em objetos.
*/