Tipos de dados e variáveis

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Antes: COMENTÁRIOS

// -> uma linha

/\*\*/ -> várias linhas

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Um sinal de igual ( = ) significa recebe.

Dois sinais de igual ( == ) significa igual.

Se uma variável receber um valor *NULL*, significa que essa variável fica desocupada.

Quando for declarar uma variável, utiliza-se a palavra “ var ”. Podemos utilizar “ let ” também.

Existem 3 formas de delimitar uma String no JS. Aspas duplas apóstrofe e crase. Vamos ver isso melhor mais tarde, mas cada uma tem uma função.

Identificadores:

- Podem começar com **letras**, **$** ou **\_**

- Não podem começar com **números**

- É possível usar **letras** e **números**

- É possível usar **acentos** e **símbolos**

- Não podem conter **espaços**

- Não podem ser **palavras reservadas**

Dicas:

- **Maiúsculas** e **minúsculas** fazem diferença

- Tente escolher **nomes coerentes** para as variáveis

Dá para utilizar o node no vs code abrindo um terminal e digitando “node”.

Para fechar o node, basta digitar “.exit” e para sair do terminal basta digitar “exit”.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TIPOS

number -> números em geral

string -> textos em geral

boolean -> verdadeiro ou falso

Para saber qual é o tipo da variável, basta utilizar o comando “ typeof ”

Interface gráfica do usuário

Descrição gerada automaticamente com confiança baixa

Texto

Descrição gerada automaticamente com confiança baixa

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Manipulação de dados

Dá para abrir uma variável usando var, let ou const. Há diferença entre eles, mas veremos depois.

Para concatenar, funciona igual Java -> usa-se o sinal de mais “ + ”

Existe o “+” para adição e “+” para concatenação

// (number + number) para adição

// (string + string) para concatenação

O window.prompt retorna um valor string.

Para resolvermos isso, precisamos fazer uma **conversão de tipos**

**String -> Número**

Number.parseInt(n) -> converte para número Inteiro

Number.parseFloat(n) -> converte para número Float

Number() -> converte para Inteiro ou Float, dependendo de que tipo de número será o resultado

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Número -> String**

String(n) -> Converte para String

n.toString() -> Converte para String

'O aluno ' + nome + ' tem ' + idade + ' anos e tirou ' + nota + ' na prova.'

- Para não precisar utilizar todas essas concatenações e ficar se confundindo e tendo dificuldade para entender o que está acontecendo, utiliza-se “template string” =>>>> ${}t

OBS: a frase vai começar com aspas.

EX: `O aluno ${nome} com ${idade} anos tirou a nota ${nota}`

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Formatando String**

Var s = ‘JavaScript’

s.length // quantos caracteres a string tem

s.toUpperCase() // deixa tudo em ‘MAIÚSCULAS’

s.toLowerCase() // deixa tudo em ‘minúsculas’

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Casas decimais e pontos ou vírgulas**

var n1 = 1545.5

n1.toFixed(n) -> serve para colocar o número (n) de casas após a vírgula ou o ponto.

n1.toFixed(n).replace(‘ . ’ , ‘ , ’) -> serve para trocar o ponto pela vírgula. Esse replace significa trocar um primeiro item por um segundo item.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Formato financeiro**

var n1 = 1545.5

n1.toLocaleString(‘pt-BR’, {style: ‘currency’, currency: ‘BRL’}) -> Real

n1.toLocaleString(‘pt-BR’, {style: ‘currency’, currency: ‘USD’}) -> Dólar

n1.toLocaleString(‘pt-BR’, {style: ‘currency’, currency: ‘EUR’}) -> Euro

Texto

Descrição gerada automaticamente

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Operadores**

- aritméticos -> 1º

- atribuição

- relacionais -> 2º

- lógicos -> 3º

- ternários

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Aritméticos**

5 + 2 = 7 -> adição

5 - 2 = 3 -> subtração

5 \* 2 = 10 -> multiplicação

5 / 2 = 2.5 -> Divisão real -> resultando sempre float

5 % 2 = 1 -> Resto da divisão

5 \*\* 2 = 25 -> Potência

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Atribuições**

**Atribuição Simples**

var a = 5 + 3 -> a=8

var b = a % 5 -> 3

var c = 5 \* b \*\* 2 -> 45

var d = 10 – a / 2 -> 6

var e = 6 \* 2 / d -> 2

var f = b % e + 4 / e -> 3

**Auto atribuição**

var n = 3

n = n + 4

n = n - 5

n = n \* 4

n = n / 2

n = n \*\* 2

n = n % 5

**Auto atribuição simplificada**

var n = 3

n = n + 4 => n+=4

n = n – 5 => n-=5

n = n \* 4 => n\*=4

n = n / 2 => n/=2

n = n \*\* 2 => n\*\*=2

n = n % 5 => n%=5

**Auto atribuição simplificada para incremento**

var n = 3

n = n + 1 ou n+=1 ou n++

n = n - 1 ou n-=1 ou n--

Obs: esse é um pós-incremento. Caso queira que seja feito antes, aí utiliza-se o pre-incremento.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Relacionais**

>

<

>=

<=

==

!=

Obs: o resultado de um operador relacional é sempre de verdadeiro ou falso

**Identidade**

5 == 5 -> TRUE

5 == ‘5’ -> TRUE

5 === ‘5’ -> FALSE

5 === 5 -> TRUE

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Lógicos**

! -> negação -> 1º

&& -> conjunção -> 2º

|| -> disjunção -> 3º

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Ternário**

?

:

Teste lógico ? true : false

Ex: media >= 7.0 ? ‘Aprovado’ : ‘Reprovado’

Obs: o funcionamento é muito parecido com o IF. A única diferença é que é mais compacto.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Ordem de precedência**

Tela de celular com texto preto sobre fundo branco

Descrição gerada automaticamente

Tela de telefone celular com letras

Descrição gerada automaticamente

**Ordem de precedência**

() -> 1º. Parênteses

\*\* -> 2º. Potência

\* / % -> 3º. Mult, div, res

+ - -> 4º. Adi, sub