**Curso de JavaScript:**

**Variáveis**:

Uma variável ser para armazenar um dado. É como um estacionamento de carro, onde cada vaga diz respeito a um tipo de veiculo que será guardado de acordo com as réguas de quem o fez. Por exemplo: var c1 = carro, var c2 = moto, var c3 = caminhão. Então toda vez que nos referirmos ao var (variável) estaremos indicando o tipo de recebimento que estamos fazendo referência. Variável pode receber números (number), letras (string), objetos e valores booleanos (boolean = false ou true).

Em JS existem três tipos de variável: var, let, const; cada uma delas com suas respectivas regras.

Como funciona uma variável:

var nome = `Gustavo`. Ao digitar nome, o programa irá mostrar o nome Gustavo.

var n = 10. Quer dizer que o n da variável equivale a 10.

**Dicas**:

Maiúsculas e minúsculas fazem diferença; tente escolher nomes coerentes para as variáveis; evite se tornar um programador alfabeto ou um programador contador.

**Identificadores**:

Podem começar com letra, $ ou \_; não podem começar com números (tipo 1n e não n1); é possível usar letras ou números; é possível também usar acentos e símbolos; não podem conter espaços; e não podem ser palavras reservadas.

**Data Types**:

Number – Infinity e NaN (not a number); string, boolean, null, undefined, objetc – array; function.

**Formatando strings**:

Var s = `JavaScript`

‘Eu estou aprendnendo s’ – // não faz interpolação.

‘Eu estou aprendendo’ + s – // usa concatenação.

‘Eu estou aprendendo ${s}’ – // usa template string.

var s = ‘JavaScript’

s.length = fala quantas caracteres a string tem.

s.toUpperCase() = transforma tudo para ‘MAIÚSCULA’.

stoLowerCase() = transforma tudo para ‘minúscula’.

**ToFixed**:

Vejamos a variável var n1 = 1545.5. Ao utilizarmos o tofixed() com um número dentro dos parênteses, iremos notar uma formatação condizente com o número da chave. Exemplo:

var n1 = 1545. 5, logo n1 vale 1545.5.

n1.toFixed(2) tornaria o valor 1545.5 em 1545.50 já que ele adiciona um número a mais depois do ponto ou virgula.

Para tornar em virgula, basta colocar: n1.toFixed(2).replace(‘.’, ‘,’):

1545,50.

toLocalString

**Operadores:**

**O que são `Operadores do JS}`**: o JS tem várias famílias de operadores, mas os principais são os “**aritméticos, atribuição, relacionados, lógicos e ternário**”.

Aritméticos são usados para fazer cálculo: (+), (-), (\*), (/), (%), (\*\*). Por exemplo: (5+2 = 7), (5\*2 = 10), (5-2 = 3), (5/2 = 2,5), (5%2 = 1) e por fim (5 \*\* 2 = 25). Cuidado: no JS quando dois operadores de conta como no exemplo “5 + 3 / 2”, que daria quatro, na linguagem de programação se faz primeiro o sinal de multiplicação ou divisão: 3/2=1,5 = 5 = 6,5. Agora, caso você queira seguir o padrão natural de 5 + 3 = 8 / 2 = 4, você vai precisar colocar (5+3) dentro de parênteses: (5+3) / 2 = 4.

**Ordem de precedência**:

A ordem com que cada um é feito: (), \*\*, \* % /e por fim + e -.

**Atribuição simples**:

var a = 5 + 3 == 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-atribuições**:

var n = 3

n = n+3 == 7: ou seja, você faz a somatória e o resultado substitui o valor original de n. Você pode ir seguindo:

n = n–5 == 2 (porque n era 7);

n = n\*4 == 8;

n = n/2 == 4;

n = n\*\*2 == 16

n = n%5 = 1

**Simplificando uma auto-atribuição**:

N = n + 4 == n+=4. Mas nesse caso, note: só é valido se a variável a receber na conta seguinte. Exemplo: var j = j + 4, simplificando seria: j+=2. Então o resto seria: n-=5, n\*=4, n/=4, n/=2, n\*\*=2, n%=5

**Incremento**:

var x = 5

x = x+1 == 6

x = x-1 == 5 simplificado: x+=1 ou x-=1

x ++ ou x--; caso eu faça ++x ou --x o programa irá somar ou subtrair antes. Exemplo:

var n = 10

n++ == 10

n == 11

n - -

11

n == 10

++n

11

--n

10.

Ainda sobre operadores: relacionais, lógicos e ternário.

Relacionais: > (maior), < (menor), >= (maior ou igual), <= (menor ou igual), == (igual), ! (diferente)=.

5 > 2 = true.

7 < 4 = false.

8 >= 8 = true.

9 <= 7 = false.

5 == 5 = true.

4 != 4 = false.

Exemplo:

preço >= 200.50 // o preço é maior ou igual a 200.50?

idade < 18 // a idade é menor do que 18?

curso == ‘JavaScript’ // a string curso é igual a JavaScript

n1 != n2 // n1 é diferente de n2?

Identidade:

5 == 5 = true.

5 == ‘5’ (valor com sinal, string) = true.

5 === ‘5’ (o mesmo valor, o mesmo tipo; no caso o 5 tem o mesmo valor de ‘5’, porém não o mesmo tipo) = false. Tem o !== desigual (que são de tipos diferentes)

Operadores lógicos:

Lógicos: ! (negação; não), && (conjunção; “e” lógico), || (disjunção; nosso “ou”).

Negação: ! só receber um valor: true ou false; ou é verdadeiro ou é falso.

Conjunção: && é um operador binário: temos dois valores lógicos de um em cada lado. Exemplo: true (caneta1) && true (caneta 2) é true porque eu recebi os dois valores que desejava; true && false = false, pois só consegui a primeira; false && true = false, pois recebi a primeira e não a segunda; false && false = false, pois não recebi nenhuma das duas. Só é satisfatório quando recebe os dois valores.

Disjunção: operador binário, pois tem valores em cada lado. E diferente da conjunção, nessa segunda basta que um deles seja verdadeiro para que o resultado seja verdadeiro: (true || true = false), (true || false = true), (false || false = false).

var a = 5

var b = 8

a > b && b % 2 == 0 (primeiro faça: operadores aritméticos, depois os operadores relacionais e por fim o operadores lógicos).

Logo: b (8) divido por 2 é 4, resto o 0 (verdadeiro); o a (5) é maior do que b (8)? Falso. *False* com *true* é *false*.

Primeiro o *não*, depois o *e* e depois o *ou*.

Exemplos:

Idade >= 15 && idade <= 17 (a idade está entre 15 e 17?)

Estado == ‘RJ’ || estado == ‘SP’ (o estado é RJ ou SP?)

Salário > 1500 && sexo != ‘M’ (o salário é acima de 1500 e não é homem?)

Ordem de precedência:

1 - (), \*\*, /...

2 - <, >, >=... (não há uma ordem correta)

3 - !, &&, ||

Operador ternário: ? e :

É o sinal de ? e : dentro de uma expressão. E é chamado de ternário por ser divido em três partes, exemplo: teste ? true : false.

Média >= 7.0 ?“aprovado”:“reprovado”