**Variáveis e Tipos primitivos**

Fazer comentários sempre para que o código fique mais legível para quem vai interpreta-lo, até para você futuramente.

* // (única linha) ou /\*\*/ (mais de uma linha) = JS
* /\*\*/ = CSS
* <!----> = HTML

**Variáveis**

Símbolo de “=” em JavaScript significa “recebe”

Memória analogia = Terreno com estacionamento p/ vagas de carros.

Memória é o local onde ficam armazenados as variáveis. (var ou let = variável)

Ex:

* var n1 = 5;
* var n2 = 8.5;
* var n3 = 15;

\*\*5, 8.5 e 15 são atributos\*\*

Assim como números (number), podem ser colocadas palavras (strings).

Ex:

* var s1 = “JavaScript”;
* var s2 = ‘Curso em Vídeo’;
* var s3 = `Guanabara`;  
    
  \*\*Podem ser usados aspas duplas, aspas simples e a crase\*\* (Possuem diferenças)

Chamamos o nome dados as variáveis de Identificadores e elas possuem regras

Regras para escolher o nome do identificar:

* Podem começar com letra, $ ou \_;
* Não podem começar com números;
* É possível usar letras ou números;
* É possível usar acentos e símbolos;
* Não podem conter espaços (substituir por underline);
* Não podem ser palavras reservadas.

Dicas:

* Maiúsculas e minúsculas fazem diferença;
* Tente escolher nomes coerentes para as variáveis;
* Evite se t ornar um “programador alfabeto” ou um “programador contador”;

**Tipos primitivos primordiais (negrito)**

* **Number;**

Infinity;

NaN.

* **String;**
* **Boolean;**
* Undefined;
* Object;

Array;

Null.

* Function.

typeof = tipo de = serve para saber o tipo da variável;

EX: var n = 200; typeof n return ‘number’

**Tratamento de Dados**

“+” serve tanto para concatenação como para adição  
  
number + number = recebe number = adição

String + string = recebe string = concatenação

Como converter tipos

**string para número**

Number.parseInt(n) ou parseInt(n) -> converter para número inteiro

Number.parseFloat(n) ou parseFloat(n) -> converter para número float (real)

Ou Number(n) para fazer os dois casos.

**número para string**

n.toString()

ou

String(n)

**Formatando Strings**

var s = ‘JavaScript’

template strings = usar `Eu sou ${nome}` ${} = placeholder

s.length = quantos caracteres a string tem

s.toUpperCase() = tudo para maiúsculas

s.toLowerCase() = tudo para minúsculas

**Operadores**

* Aritméticos;
* Atribuição;
* Relacionais;
* Lógicos;
* Ternários;

Obs: existem outras famílias de operadores (ex: typeof() )

**Aritméticos**

Adição = +

EX: 5 + 2 = 7

Subtração = -

EX: 5 - 2 = 3

Multiplicação = \*

EX: 5 \* 2 = 10

Divisão = /

EX: 5 / 2 = 2.5

Divisão inteira (resto) = %

EX: 5 % 2 = 1

Exponenciação = \*\*

EX: 5 \*\* 2 = 25

**Ordem de precedência de operadores aritméticos**

EX: 5 + 3 / 2 = 6.5 (5 + 3) / 2 = 4

1° - () = parênteses;

2° - \*\* = potência;

3° - / % \* = multiplicação, divisão e divisão inteira

4° - + - = soma e divisão

**Atribuição**

**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; (simples)

n = n + 4 = 7; para simplificar n += 4

n = n – 5 = 2; n -= 5

n = n \* 4 = 8; n \*= 4

n = n / 2 = 4; n /= 4

n = n \*\* 2 = 16; n \*\*= 2

n = n % 5 = 1; n %= 5

**Incremento**

Var x = 5;

x = x + 1 = 6; x += 1 x ++

x = x -1 = 5; x -= 1 x—

**Relacionais**

Maior = >

EX: 5 >3 = true

Menor = <

EX: 7 < 4 = false

Maior igual = >=

EX: 8 >= 8 = true

Menor igual = <=

EX: 9 <= 7 = false

Igualdade = “==”

EX: 5 == 5 = true

Não igual = diferente = “!=”

EX: 4 != 4 = false

**Identidade**

5 == 5 = true

5 == ‘5’ = true, mesmo valor mas tipo diferente == testa o valor

Operador de igualdade restrita

5 === ‘5’ = false, mesmo valor e mesmo tipo

5 != ‘5’ = false

5 !== ‘5’ = true (desigual restrito)

**Lógicos**

! = negação = “não” lógico

Operador unário (só tem 1 operando) = true or false

!true = false

!false = true

&& = conjunção = “e” lógico

Operador binário (2 valores lógicos) =

True && true = true

True && false = false

False && true = false

False && false = false

|| = disjunção = “ou” lógico

Operador binário (2 valores lógicos)

True || true = true

True || false = true

False || true = true

False || false = false

Ordem de precedência dos operadores

1° aritméticos

1° - () = parênteses;

° - \*\* = potência;

3° - / % \* = multiplicação, divisão e divisão inteira

4° - + - = soma e divisão

2° relacionais

Não tem ordem de precedência, quem aparecer primeiro na ordem da esquerda para a direita.

3° lógicos

Ordem de precedência dos operadores lógicos

1° ! (negação)

2° && (conjunção)

3° || (disjunção)

4° ternário

**Operador ternário**

Interrogação (?) e dois pontos (:)

Possuem 3 blocos

teste lógico ? true : false

média >= 7.0 ? “Aprovado” : “Reprovado”