**JavaScript**

**Comentários:**

* // - uma única linha de código;
* /\* conteúdo \*/ - mais de uma linha de código.

**Simbolos:**

* = : recebe;
* + : concatenação;
  + (number + number) para adição;
  + (string + string) para concatenação;

**Variáveis**:

No JavaScript moderno, além de utilizar a palavra **var**, também podemos utilizar a palavra **let**.

* var a1 = 200: var a1 **recebe** 200;
* let a1 = 100: let a1 **recebe** 100.

**Identificadores**:

* 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**;
* Não podem ser **palavras** **reservadas** (palavras que o JavaScript usa como comandos);
* Dicas:
  + **Maiúsculas** e **minúsculas** fazem diferença;
  + Tente utilizar **nomes** **coerentes** para as variáveis;
  + Evite se tornar um ‘**programador** **alfabeto’** ou ‘**programador** **contador’**;

**Data** **Types:**

* number: números em geral;
* string: caracteres entre “, ‘ ou `:
* boolean:
* null:
* undefined:
* object: entre [] ou {};
* function: function() {}

**Conversão:**

* String para número:
  + Number.parseInt(n) – converter para inteiro;
  + Number.parseFloat(n) – converter para real;
  + Number(n) – converter para qualquer tipo de número.
* Número para string:
  + String(n) – o que está entre parênteses se transforma em uma string;
  + n.toString() – o que você quer converter para uma string vem primeiro;

**Formatando Strings:**

**var** **s** = ‘JavaScript’

* ‘Estou aprendendo **s**’ – não faz interpolação;
* ‘Estou aprendendo’ **+** **s** – usa concatenação;
* `Estou aprendendo **${s}**` - usa template string;
* s.**length** – monstra quantos caracteres a string tem;
* s.**toUpperCase()** – muda tudo para MAIÚSCULAS;
* s.**toLowerCase()** – muda tudo para minúsculas;

**Formatando Números:**

**var** **n1** = 1543.5

* n1.**toFixed(**2**)** – muda para 2 casas decimais;
* n1.**toReplace(**‘.’, ‘,’**)** – troca uma coisa por outra;
* n1.**toLocaleString(**‘pt-BR’, {style: ‘currency’, currency: ‘BRL’}**)** – transforma em valor monetário;

**Operadores:**

* aritméticos:

(A = 5)

(B = 2)

|  |  |  |
| --- | --- | --- |
| + | Adição | A + B = 7 |
| - | Subtração | A – B = 3 |
| \* | Multiplicação | A \* B = 10 |
| / | Divisão | A / B = 2.5 |
| \*\* | Exponenciação | A \*\* B = 25 |
| % | Módulo (resto de uma divisão) | A % B: 1 |

Obs: divisão utiliza tipo: real.

* atribuição:

Atribuições simples:

|  |  |
| --- | --- |
| var a = 5 + 3 | var a = 8 |
| var b = a % 5 | var b = 3 |
| var c = 5 \* b \*\* 2 | var c = 45 |
| var d = 10 – a / 2 | var d = 6 |
| var e = 6 \* 2 / d | var e = 2 |
| var f = b % 3 + 4 / e | var f = 3 |

Auto atribuições:

**var** **n** = 3

|  |  |  |
| --- | --- | --- |
| n += 4 | n = n + 4 | n = 7 |
| n -= 5 | n = n – 5 | n = 2 |
| n \*= 4 | n = n \* 4 | n = 8 |
| n /= 2 | n = n / 2 | n = 4 |
| n \*\*= 2 | n = n \*\* 2 | n = 16 |
| n %= 5 | n = n % 5 | n = 1 |

Incremento:

var x = 5

|  |  |  |
| --- | --- | --- |
| x = x + 1 | x ++ | x = 6 |
| x = x - 1 | x -- | x = 5 |

* relacionais:

|  |  |
| --- | --- |
| > | Maior que |
| < | Menor que |
| >= | Maior ou igual a |
| <= | Menor ou igual a |
| == | Igual a |
| === | Idêntico a (mesmo valor e tipo) |
| != | Diferente de |
| !== | Desigual restrito a (mesmo valor e tipo) |

* lógicos:
  + && (E): os dois precisam ser verdadeiros para o resultado ser verdadeiro;

|  |  |  |
| --- | --- | --- |
| **p** | **q** | **p && q** |
| V | V | V |
| V | F | F |
| F | V | F |
| F | F | F |

* + || (OU): algum ou os dois precisam ser verdadeiros para o resultado ser verdadeiro;

|  |  |  |
| --- | --- | --- |
| **p** | **q** | **p || q** |
| V | V | V |
| V | F | V |
| F | V | V |
| F | F | F |

* + ! (NÃO): ocorre a inversão do resultado lógico;

|  |  |
| --- | --- |
| **p** | **! p** |
| V | F |
| F | V |

* ternário: ? :

**teste** **lógico** **?** **true** (o que vai acontecer quando o teste for true) **:** **false** (o que vai acontecer quando o teste for false).

**Ordem de precedência geral de operadores:**

|  |  |
| --- | --- |
| Aritméticos | 1º ( ) |
| 2º \*\* |
| 3º \* / % |
| 4º + - |
| Relacionais | 5º Todos iguais |
| Lógicos | 6º ! (NÃO) |
| 7º && (E) |
| 8º || (OU) |

**Selecionando elementos para navegar na área DOM:**

* por Marca:
  + getElementsByTagName() – consegue selecionar mais de um objeto; [] seleciona qual elemento de acordo com o número dentro da chave.
* por ID:
  + getElementByID()
* por Nome:
  + getElementsByName()
* por Classe:
  + getElementsByClassName()
* por Seletor:
  + querySelector() – usa a sintaxe do css
  + querySelectorAll() - usa a sintaxe do css

**Funções:**

function ação (param ) {

bloco

}

addEventListener(‘evento DOM’, função) – executa a função e deixa o html mais limpo .

**Eventos DOM:**

* mouseenter;
* mousemove;
* mousedown;
* mouseup;
* click;
* mouseout;
* createElement() – cria um elemento em HTML;
* setAttribute() – adiciona um atributo;
* appendChild() – adiciona elemento;

**Condições:**

* if (condição simples):

**if (**condição**)** **{**

true

**}**

* if (condição composta):

**if (**condição**)** **{**

true

**} else** **{**

false

**}**

* if (condição aninhada):

**if(**cond1**)** **{**

bloco1

**}** **else** **If(**cond2**)** **{**

bloco2

**} else {**

bloco3

**}**

* switch (condição múltipla): é obrigatório usar o break

**switch(**expressão**)** **{**

**case** valor 1**:**

break

**case** valor 2**:**

break

**case** valor 3**:**

break

**default:**

break

**}**

**Repetições:**

* while (enquanto), primeiro testa e depois executa o bloco:

**while(**condição**)** **{**

bloco A

**}**

* do while (faça enquanto), primeiro executa o bloco e depois testa:

**do{**

bloco A

**}** **while(**condição**)**

* for (para):

**for(**inicio ; teste ; incremento**)** **{**

bloco A

**}**