Biblioteca MATH

> Conjunto de diversas operações matemáticas para uso diversificado envolvendo cálculos. Ela possui métodos e propriedades, sendo possível fazer aplicações que envolvem todo tipo de cálculo, seja ele trigonometria, logaritmo, operações básicas, módulos e etc.

A biblioteca Math é utilizada em diversas aplicações pela Web e, por isso, é de extrema importância que a conheçamos. Com seus métodos e propriedades, nos ajuda a construir expressões matemáticas mais rapidamente, além de nos oferecer várias possiblidades de uso com precisão total dos números reais em Javascript.

Métodos → São os meios disponibilizados para fazer algum cálculo ou expressão qualquer.

Propriedades → São números já 'definidos' que usamos em outro cálculo.

Os métodos recebem como parâmetro valores com algorismos, porém, caso use algumas exceções não irá funcionar. Essas exceções retornarão NaN como valor:

```
Math.metodoQualquer([1, 2]);  // Array com mais de 1 item
Math.metodoQualquer({});      // Objeto
Math.metodoQualquer('texto'); // String com letras
Math.metodoQualquer();        // Vazio
```

Seus métodos são:

- Math.abs(x) → Retorna o módulo de um número qualquer, ou seja, seu valor absoluto. É responsável por nos trazer a distância de determinado número até o 0.

  Exemplos:

  |10| = 10 → pois há 10 unidades até o 0;

  |-5| = 5 → pois há 5 unidades até o 0;

  → Exemplos de uso:

```
Math.abs('-10');    // 10
Math.abs(-2);       // 2
Math.abs(null);     // 0
Math.abs('');       // 0
Math.abs([]);       // 0
```

```
Math.abs([2]);      // 2
Math.abs([1,2]);    // NaN
Math.abs({});       // NaN
Math.abs('string'); // NaN
Math.abs();         // NaN
```

Devemos inserir um determinado valor como parâmetro, caso contrário, nos retornará NaN. Esse valor deve conter algarismos em geral e, caso seja uma string, só funcionará se possuir apenas números ('10' → '11' → '-9').

- Math.floor(x) → Retorna o maior inteiro que é igual ou menor a um número, ou seja, ele irá calcular qual é o inteiro mais próximo a partir de um número (passado por parâmetro).

  Exemplo:

  5.1 → Maior inteiro = 5;

  -5.9 → Maior inteiro = -6;

  -5.1 → Maior inteiro = -6;

```
Math.floor(-5.1)    // -6
Math.floor(-5.9)    // -6
Math.floor(5.1)     // 5
Math.floor('')      // 0
Math.floor([])      // 0
Math.floor(null)    // 0
Math.floor([1])     // 1
Math.floor('1')     // 1
Math.floor('texto') // NaN
Math.floor()        // NaN
Math.floor([1, 2])  // NaN
Math.floor({})      // Nan
```

- Math.log10(x) → Retorna o logaritmo de base 10 de um número. É passado o número em questão por parâmetro e o método é responsável por calcular seu logaritmo considerando sua base como 10.
  - Antes de prosseguirmos à codificação de exemplo, devemos saber o que é um Logaritmo e como calculá-lo.
  - Logaritmo:

    Um logaritmo é usado em áreas como a matemática, computação, ciência e outros, a fim de facilitar contas diversas em nossas atividades.

    Sua nomenclatura é:

$$\log_a b = x$$

    Onde:

    - a → base;
    - b → logaritmando;
    - x → logaritmo;

Quando possuímos um logaritmo de base decimal (base 10), não é necessário escrevermos a mesma, basta inserirmos a palavra 'log' e o valor desejado:

$$\log 100 = 2$$

Ao resolvermos um logaritmo de base 10 devemos pensar quantos '10s' conseguimos multiplicar para chegar ao logaritmando. Por exemplo:

log 1000 = x;

"Quantos 10s devo multiplicar para chegar a 1000?"

E então a resposta deve ser x = 3, pois 10 x 10 x 10 = 1000.

Podemos também compará-lo à maneira exponencial:

$$\log_a b = x \quad <=> \quad a^x = b$$

- Exemplo de uso:

```
Math.log10(100);   // 2
Math.log10(1000);  // 3
```

- Math.max(x, y, z, ..., a) → Retorna o maior valor dentre os parâmetros recebidos:

```
Math.max(1, 2, 5, 0);       // 5
Math.max(1, 2, 5, 10);      // 10
Math.max(5, 2, 3, 0);       // 5
Math.max(-1, 2, -5, -10);   // 2
```

- Math.min(x, y, z, ..., a) → Retorna o menor valor dentre os parâmetros recebidos:

```
Math.min(1, 2, 5, 0);       // 0
Math.min(1, 2, 5, 10);      // 1
Math.min(5, 2, 3, 0);       // 0
Math.min(-1, 2, -5, -10);   // -10
```

- Math.pow(x, y) → Retorna a base (x) elevada ao expoente (y). Para usarmos esse método devemos passar dois valores para que ocorra o cálculo desejado.
  - Para prosseguirmos devemos saber sobre exponenciação e como calculamos tal.
  - Exponenciação:

    É uma operação matemática muito usada, responsável por facilitar a escrita de um valor com diversos algorismos. Em seu cálculo é utilizado dois valores: a base e o expoente.

    Sua nomenclatura é:

$$base^{expoente}$$

    Exemplo:

$$2^2$$

    Para resolvermos precisamos pegar a base e multiplicarmos pela quantidade do expoente: 2 x 2 = 4
  - Há várias possibilidades de expoentes em geral, como:
    - Simples:

$$2^2$$

    - Fracionários:

$$2^{1/3}$$

    - Negativos:

$$2^{-2}$$

  - Também há uma possibilidade de base negativa:

→ Base negativa: se o expoente for par, o resultado será par, caso seja ímpar o resultado será ímpar.

$$-2^2$$

- Exemplos de uso:

```
/* Expoente simples: */
Math.pow(10, 2); // 100
Math.pow(5, 3);  //125

/* Expoente fracionário: */
Math.pow(3, 0.5);  // 1.7320508075688772 (raiz quadrada de 3)
Math.pow(8, 1/3);  // 2 (raiz cúbica de 8)
Math.pow(16, 0.5); // 4 (raiz quadrada de 16)

/* Expoentes negativos: */
Math.pow(2, -2);    // 0.25 <=> (1/2)^2 <=> 1/4
Math.pow(8, -1/3);  // 0.5 <=> (1/8)^1/3 <=> 1/2

/* Base negativa: */
Math.pow(-8, 2);   // 64
Math.pow(-8, 3);   // -512
Math.pow(-8, -3);  // -0.001953125

/*
Por conta de raízes "par" e "ímpar" serem próximas, limitando a precisão
do ponto flutuante, bases negativas com expoente fracionário retornam NaN.
*/

Math.pow(-2, 1/3)  // NaN
```

- Math.random() → Retorna um número aleatório entre 0 e 1. Esse método nos traz um número decimal aleatório, incluindo o 0 e excluindo o 1, ou seja, um número menor que 1. Para inteiros aleatórios podemos utilizar uma combinação de métodos, misturando o Math.floor() com o Math.random():
  - `Math.floor(Math.random() * 10);` → nos retorna algum inteiro entre 0 e 9, pois escolhe um número aleatório entre 0 e 1 (.random), multiplica por 10 e por fim, retorna o maior inteiro mais próximo (.floor).
  - Exemplo de uso:

```
Math.random();  // 0.8690168776595018
Math.random();  // 0.7832314709660295
Math.random();  // 0.0676909712495184

/* Para obtermos um número aleatório entre 0 e 10: */
Math.floor(Math.random() * 11) // 10
Math.floor(Math.random() * 11) // 3
Math.floor(Math.random() * 11) // 8
```

> Assim como possuímos as chamadas funções trigonométricas, temos suas funções recíprocas. Portanto, é possível calcularmos o arco de seno (Math.asin(x)), arco de cosseno (math.acos(x)) e arco tangente (Math.atan(x)).

- Math.acos(x) → Retorna o arco cosseno de um número. A chamada função 'arcos', ou função inversa da função cosseno, permite esse cálculo. Ele retornará um valor entre 0 e π radianos para parâmetros entre -1 e 1, caso esteja fora desse escopo, retornará NaN.
  - Exemplos de uso:

```
Math.acos(1);   // 0
Math.acos(0.5); // 1.0471975511965979
Math.acos(0.4)  // 1.1592794807274085
Math.acos(0);   // 1.5707963267948966
Math.acos(-1);  // 3.141592653589793

Math.acos(2);   // NaN
Math.acos(-2);  // NaN
// *Valores acima de 1 ou menores que -1 retornam NaN
```

- Math.acosh(x) → Retorna o arco de cosseno hiperbólico de um número. É muito usada em engenharias, onde calcula, por exemplo, o transporte elétrico, superestrutura e aeroespacial. Devemos usá-la para valores maior que 1, caso contrário, retornará NaN:
  - Exemplos de uso:

```
Math.acosh(2);   // 1.3169578969248166
Math.acosh(1);   // 0
Math.acosh(0);   // NaN
Math.acosh(0.5); // NaN
Math.acosh(-1);  // NaN

// *Valores abaixo de 1 retornam NaN
```

- Math.ains(x) → Retorna o arco seno de um número. Ele trará um valor em radianos para parâmetros entre -1 e 1, caso esteja fora desse escopo, retornará NaN.
  - Exemplos de uso:

```
Math.asin(1);   // 1.5707963267948966 (π/2)
Math.asin(0.5); // 0.5235987755982989
Math.asin(0);   // 0
Math.asin(-1);  // -1.5707963267948966 (-π/2)

Math.asin(2);   // NaN
Math.asin(-2);  // NaN
// *Valores acima de 1 ou menores que -1 retornam NaN
```

- Math.asinh(x) → Retorna o arco seno hiperbólico de um número. Assim como a função arco tangente, foi desenvolvida para resoluções de integrais.
  - Exemplos de uso:

```
Math.asinh(2);   // 1.4436354751788103
Math.asinh(1);   // 0.881373587019543
Math.asinh(0.5); // 0.48121182505960347
Math.asinh(0);   // 0
Math.asinh(-1);  // -0.881373587019543
```

- Math.atan(x) → Retorna o arco tangente de um número. Ele trará um valor entre -π/2 e π/2 radianos.

  * Caso queira evitar usar +- Infinity, utilize Math.atan2() com segundo parâmetro igual a 0.
  - Exemplos de uso:

```
Math.atan(1);   // 0.7853981633974483
Math.atan(0);   // 0
Math.atan(-0);  // -0

Math.atan(Infinity);   //  1.5707963267948966
Math.atan(-Infinity);  // -1.5707963267948966

/* O ângulo que a linha [(0,0); (x, y)] forma com o eixo x em um sistema
de coordenadas cartesianas: */
Math.atan(y / x);
```

- Math.atan2(y, x) → Retorna o arco tangente do coeficiente dos argumentos passados. Onde 'y' é o primeiro número e 'x' é o segundo. O valor retornado é entre -π e π, representando o ângulo teta entre x e y.
  - Exemplos de uso:

```
Math.atan2(90, 15); // 1.4056476493802699
Math.atan2(15, 90); // 0.16514867741462683

Math.atan2(±0, -0);              // ±π.
Math.atan2(±0, +0);              // ±0.
Math.atan2(±0, -x);              // ±π para x > 0.
Math.atan2(±0, x);               // ±0 para x > 0.
Math.atan2(-y, ±0);              // -π/2 para y > 0.
Math.atan2(y, ±0);               // π/2 para y > 0.
Math.atan2(±y, -Infinity);       // ±π para finito y > 0.
```

```
Math.atan2(±y, +Infinity);        // ±0 para finito y > 0.
Math.atan2(±Infinity, x);         // ±π/2 para finito x.
Math.atan2(±Infinity, -Infinity); // ±3*π/4.
Math.atan2(±Infinity, +Infinity); // ±π/4.
```

- Math.atanh(x) → Retorna o arco tangente hiperbólico de um número. Para valores maiores que 1 ou menores que -1, é retornado NaN.
  - Exemplos de uso:

```
Math.atanh(0);   // 0
Math.atanh(0.5); // 0.5493061443340548

Math.atanh(2);   // NaN
Math.atanh(-2);  // NaN
Math.atanh(1);   // Infinity
Math.atanh(-1);  // -Infinity
```

- Math.cbrt(x) → Retorna a raiz cúbica de um número.

  Calculamos uma raiz cúbica pensando em quantas vezes aquele valor é multiplicado. Por exemplo:

  $\sqrt[3]{8}$ = ?

  "Qual número multiplicado por si 3 vezes é igual a 8?"

  Se a resposta foi 2 você acertou!

  $\sqrt[3]{8}$ = 2, pois 2 x 2 x 2 = 8;

  - Exemplos de uso:

```
Math.cbrt(8)          // 3
Math.cbrt(2);         // 1.2599210498948734
Math.cbrt(1);         // 1
Math.cbrt(null);      // 0
Math.cbrt(0);         // 0
Math.cbrt(-0);        // -0
Math.cbrt(-1);        // -1

Math.cbrt(NaN);       // NaN

Math.cbrt(Infinity);  // Infinity
Math.cbrt(-Infinity); // -Infinity
```

- Math.ceil(x) → Retorna o menor inteiro que é igual ou maior a um número, ou seja, ele irá calcular qual é o inteiro mais próximo a partir de um número (passado por parâmetro).

  Menor inteiro:

  0.95 → 1

  7.004 → 8

  10 → 10

  - Exemplos de uso:

```
Math.ceil(.95);    // 1
Math.ceil(4);      // 4
Math.ceil(7.004);  // 8
Math.ceil(-0.95);  // -0
Math.ceil(-4);     // -4
Math.ceil(-7.004); // -7
```

- Math.cos(x) → Retorna o cosseno de um ângulo, que deve estar em radianos. Retorna um valor entre -1 e 1.

  Como transformar ângulo em radiano:

  $$Angulos * \pi / 180$$

  Exemplos:

  120º → 120 * π / 180 = 2 / 3 rad

30º → 30 * π / 180 = 1 / 6 rad

225º → 225 * π / 180 = 5 / 4 rad

- Exemplos de uso:

```
Math.cos(0);               // 1
Math.cos(2 * Math.PI);     // 1
Math.cos((1/6) * Math.PI); // 0.8660254037844387
Math.cos(1);               // 0.5403023058681398
Math.cos(Math.PI);         // -1
```

- Math.cosh(x) → Retorna o cosseno hiberbólico de um número.
  - Exemplos de uso:

```
Math.cosh(0);  // 1
Math.cosh(2);  // 3.7621956910836314
Math.cosh(1);  // 1.5430806348152437
Math.cosh(-1); // 1.5430806348152437
Math.cosh(-2); // 3.7621956910836314
```

- Math.exp(x) → Retorna e^x, onde x é o argumento e e é a Constante de Euler, a base dos logaritmos naturais.

  Quando falamos sobre logaritmos naturais, significa que é o logaritmo daquele determinado número na base igual a 2.71, ou seja, na base e.

  Constante de Euler = 2.718281828459045;

  - Exemplos de uso:

```
Math.exp(2);  // 7.38905609893065
Math.exp(1);  // 2.718281828459045
Math.exp(0);  // 1
Math.exp(-1); // 0.36787944117144233
```

- Math.expm1(x) → Retorna e^x - 1, onde x é o argumento e e é a Constante de Euler, a base dos logaritmos naturais.
  - Exemplos de uso:

```
Math.expm1(2)   // 6.38905609893065
Math.expm1(1);  // 1.718281828459045
Math.expm1(0);  // 0
Math.expm1(-1); // -0.6321205588285577
```

- Math.fround(x) → Retorna o mais próximo ponto de precisão simples de 32 bits do número fornecido.

JavaScript usa números de ponto flutuante duplo de 64 bits, oferecendo uma alta precisão. Porém, há casos que você poderá trabalhar com números de ponto flutuante de 32 bits e, caso haja a verificação de igualdade entre ambos (32 bits e 64 bits), acarretará em problemas. Caso o número em questão estiver fora do intervalo suportado pelo de 32 bits, será retornado ±Infinity.

- Exemplos de uso:

```
// Ao compararmos o número 1.5 por exemplo, veremos que ambos (32bits e 64bits)
// são iguais:
Math.fround(1.5); // 1.5
Math.fround(1.5) === 1.5; // true

// Porém, ao compararmos o número 1.337 já vemos a diferença:
Math.fround(1.337); // 1.3370000123977661
Math.fround(1.337) === 1.337; // false

// Já quando possuímos um valor fora do intervalo suportado, nos é
// retornado Infinity:
2 ** 150; // 1.42724769270596e+45
Math.fround(2 ** 150); // Infinity

// Caso o parâmetro não consiga ser convertido, será retornado NaN:
Math.fround('Faculdade iv2'); // NaN
Math.fround(NaN); // NaN
```

- Math.hypot([x[, y[, ...]]]) → Retorna a raiz quadrada do somatório do quadrado dos parâmetros recebidos. Ou seja, ela calcula o quadrado de seus parâmetros e nos traz sua raiz quadrada (hipotenusa).

  Em vez de usarmos: Math.sqrt(v1 * *v1 + v2 * *v2), podemos simplesmente usar Math.hypot(v1, v2). Caso nenhum parâmetro seja passado, retorna 0, caso algum parâmetro não possa ser convertido para o tipo number, o resultado será NaN.

  - Exemplos de uso:

```
Math.hypot(3, 4, 5);        // 7.0710678118654755
Math.hypot(3, 4, '5');      // 7.0710678118654755 ('5' -> 5)
Math.hypot(3, 4);           // 5
Math.hypot();               // 0
Math.hypot(NaN);            // NaN
Math.hypot(3, 4, 'teste');  // NaN ('teste' -> NaN)

Math.hypot(-3);             // 3
// Caso seja passado somente um parâmetro, terá a mesma função que
// Math.abs()
```

- Math.imul(x, y) → Retorna uma multiplicação de 32 bits dos dois parâmetros recebidos, semelhante a C.

  - Exemplos de uso:

```
Math.imul(2, 4);          // 8
Math.imul(-1, 8);         // -8
Math.imul(-2, -2);        // 4
Math.imul(0xffffffff, 5); // -5
Math.imul(0xfffffffe, 5); // -10
```

- Math.log(x) → Retorna o logaritmo natural de um número dado. Caso o número seja negativo, o resultado será NaN.

  - Exemplos de uso:

```
Math.log(10); // 2.302585092994046
Math.log(5);  // 1.6094379124341003
Math.log(1);  // 0
Math.log(0);  // -Infinity
Math.log(-1); // NaN
```

- Math.log2() → Retorna o logaritmo de base 2 de um número dado. Caso o número seja negativo, o resultado será NaN. Essa função é equivalente a Math.log(x) / Math.log(2).

  - Exemplos de uso:

```
Math.log2(1024); // 10
Math.log2(3);    // 1.584962500721156
Math.log2(2);    // 1
Math.log2(1);    // 0
Math.log2(0);    // -Infinity
Math.log2(-2);   // NaN
```

- Math.log1p(x) → Retorna o logaritmo natural de 1 + o número fornecido. Caso o número seja menor que -1, o resultado será NaN. Usamos esse método quando x é um número muito pequeno, por exemplo 1^-15.

  - Exemplos de uso:

```
Math.log1p(1);  // 0.6931471805599453
Math.log1p(0);  // 0
Math.log1p(-1); // -Infinity
Math.log1p(-2); // NaN
```

- Math.round(x) → Retorna o valor de um número arredondado para o inteiro mais próximo. Um exemplo de parâmetro que podemos ter é '0.5'. Caso tenha um valor maior que esse, será arredondado para o inteiro maior. Caso seja menor, será arredondado para o inteiro menor.

  - Exemplos de uso:

```
Math.round(42);     // 42
Math.round(20.49);  // 20
Math.round(20.5);   // 21
Math.round(5.95);   // 6
Math.round(5.5);    // 6
Math.round(5.05);   // 5
Math.round(-20.5 ); // -20
Math.round(-20.51); // -21
```

- Math.sign(x) → Retorna 1 positivo ou 1 negativo. Ele recebe o parâmetro e retorna se o mesmo é positivo (maior que 0), 0 ou negativo (menor que 0).
  - Exemplos de uso:

```
Math.sign(3);       //  1
Math.sign(-3);      // -1
Math.sign('-3');    // -1
Math.sign(0);       //  0
Math.sign(-0);      // -0
Math.sign(NaN);     // NaN
Math.sign('Facul'); // NaN
Math.sign();        // NaN
```

- Math.sin(x) → Retorna o seno de um ângulo, que deve estar em radianos. Retorna um valor entre -1 e 1.
  - Exemplos de uso:

```
Math.sin(10);          // -0.5440211108893698
Math.sin(1);           // 0.8414709848078965
Math.sin(0);           // 0
Math.sin(-10);         // 0.5440211108893698
Math.sin(Math.PI / 2); // 1
```

- Math.sihn(x) → Retorna o arco de seno hiperbólico de um número. É usado na área da Estatística, onde atua na transformação de Johnson para transformar dados de modo a seguir uma distriuição normal.
  - Exemplos de uso:

```
Math.sinh(2);  // 3.626860407847019
Math.sinh(1);  // 1.1752011936438014
Math.sinh(0);  // 0
Math.sinh(-1); // -1.1752011936438014
```

- Math.sqrt(x) → Retorna a raiz quadrada de um número. Para calcularmos a raiz quadrada de determinado número devemos pensar qual número multiplicado por si mesmo é igual ao radicando.

$$\sqrt[n]{a} = b <=> b^n = a$$

  - Onde:
    - n: índice;
    - a: radicando;
    - √: radical;
  - Exemplos:
    - √121 = 11, pois 11 x 11 = 121;
    - √49 = 7, pois 7 x 7 = 49;
  - Exemplos de uso:

```
Math.sqrt(16);  // 4
Math.sqrt(9);   // 3
Math.sqrt(2);   // 1.414213562373095
Math.sqrt(1);   // 1
```

```
Math.sqrt(0);   // 0
Math.sqrt(-1);  // NaN
Math.sqrt(-0);  // -0
```

- Math.tan(x) → Retorna a tangente de um ângulo, que deve estar em radianos.
  - Exemplos de uso:

```
Math.tan(45); // 1.6197751905438615
Math.tan(1);  // 1.5574077246549023
Math.tan(10); // 0.6483608274590866
```

- Math.tanh(x) → Retorna a tangente hiberbólica de um número. A tângente hiberbólica é obtida pela razão entre o seno hiberbólico e o cosseno hiberbólico.
  - Exemplos de uso:

```
Math.tanh(Infinity); // 1
Math.tanh(10);       // 0.9999999958776927
Math.tanh(1);        // 0.7615941559557649
Math.tanh(0);        // 0
Math.tanh(-10);      // -0.9999999958776927
```

- Math.trunc(x) → Retorna a parte inteira de um número removendo quaisquer dígitos fracionários. Aceita valores positivos e negativos, seu trabalho é apenas "cortar" as casas decimais e retornar a parte inteira:
  - Exemplos de uso:

```
Math.trunc(42.84);      // 42
Math.trunc(13.37);      // 13
Math.trunc(0.123);      //  0
Math.trunc(-0.123);     // -0
Math.trunc('-1.123');   // -1
Math.trunc(NaN);        // NaN
Math.trunc('Facul');    // NaN
Math.trunc();           // NaN
```

Suas propriedades são:

- Math.PI → O número PI (3.14159...), representação da razão entre a circunferência de um círculo e seu diâmetro.
  - Exemplo de uso:

```
//Constante chamada 'num8' que recebe a propriedade do PI
const num8 = Math.PI;

console.log(num8);
//Resultado = 3.141592653589793
```

- Math.SQRT2 → A raiz quadrada de 2, aproximadamente 1,414...
  - Exemplo de uso:

```
//Constante chamada 'num9' que recebe a propriedade da raiz quadrada de 2
const num9 = Math.SQRT2;

console.log(num9);
//Resultado = 1.4142135623730951
```

- Math.E → Constante de Euler e base dos logaritmos naturais, aproximadamente 2,718...
  - Exemplo de uso:

```
//Constante chamada 'num10' que recebe a propriedade da constante de Euler
const num10 = Math.E;

console.log(num10);
//Resultado = 2.718281828459045
```

- Math.LN2 → Logaritmo natural de 2, aproximadamente 0,693...
  - Exemplo de uso

```
//Constante chamada 'num11' que recebe a propriedade do logaritmo natural de 2
const num11 = Math.LN2;

console.log(num11);
//Resultado = 0.6931471805599453
```

- Math.LN10 → Logaritmo natural de 10, aproximadamente 2.302...
  - Exemplo de uso:

```
//Constante chamada 'num12' que recebe a propriedade do logaritmo natural de 10
const num12 = Math.LN10;

console.log(num12);
//Resultado = 2.302585092994046
```

- Math.LOG2E → Logaritmo de base 2 de e, aproximadamente 1,442...
  - Exemplos de uso:

```
//Constante chamada 'num13' que recebe a propriedade do logaritmo de base 2 de e
const num13 = Math.LOG2E;

console.log(num13);
//Resultado = 1.4426950408889634
```

- Math.LOG10E → Logaritmo de base 10 de e, aproximadamente 0,434...
  - Exemplos de uso:

```
//Constante chamada 'num14' que recebe a propriedade do logaritmo de base 10 de e
const num14 = Math.LOG10E;

console.log(num14);
//Resultado = 0.4342944819032518
```

- Math.SQRT1_2 → Raiz quadrada de 1/2, aproximadamente 0,707...
  - Exemplos de uso:

```
//Constante chamada 'num15' que recebe a propriedade da raiz quadrada de 1/2
const num15 = Math.SQRT1_2;

console.log(num15);
//Resultado = 0.7071067811865476
```

Bibliografias:

- https://www.w3schools.com/js/js_math.asp
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Math