# Números de JavaScript

[❮ Anterior](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/js_string_methods.asp&usg=ALkJrhgcCr7Qk9SXEMOiLjz6ozwBn7m0LQ)[Próximo ❯](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/js_number_methods.asp&usg=ALkJrhhLW5r1rthV-07AzrRkSCufdl_KTQ)

O JavaScript tem apenas um tipo de número. Os números podem ser escritos com ou sem decimais.

### Exemplo

var x = 3.14;    // A number with decimals  
var y = 3;       // A number without decimals

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers1&usg=ALkJrhiXvGigkdcO4K_V8u_z2kVyBpzRqw)

Os números extra grandes ou extra pequenos podem ser escritos com notação científica (expoente):

### Exemplo

var x = 123e5;    // 12300000  
var y = 123e-5;   // 0.00123

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers2&usg=ALkJrhj1sFsIAH8c2UvLpTRc5en6HcssDg)

## Os números de JavaScript são sempre um ponto flutuante de 64 bits

Ao contrário de muitas outras linguagens de programação, o JavaScript não define diferentes tipos de números, como números inteiros, curto, longo, ponto flutuante etc.

Os números de JavaScript são sempre armazenados como números de ponto flutuante de dupla precisão, seguindo o padrão internacional IEEE 754.   
  
Este formato armazena números em 64 bits, onde o número (a fração) é armazenado nos bits 0 a 51, o expoente nos bits 52 a 62 e o bit de registro 63:

|  |  |  |
| --- | --- | --- |
| **Valor (aka Fraction / Mantissa)** | **Expoente** | **Placa** |
| 52 bits (0 - 51) | 11 bits (52 - 62) | 1 bit (63) |

## Precisão

Inteiros (números sem um período ou notação de expoente) são precisos até 15 dígitos.

### Exemplo

var x = 999999999999999;   // x will be 999999999999999  
var y = 9999999999999999;  // y will be 10000000000000000

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_inaccurate1&usg=ALkJrhhQXbMIQROBV2VemzFWxRw3MA4Nrw)

O número máximo de decimais é de 17, mas a aritmética de ponto flutuante nem sempre é 100% precisa:

### Exemplo

var x = 0.2 + 0.1;         // x will be 0.30000000000000004

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_inaccurate2&usg=ALkJrhj6cF2JO4WE7tvhi7TxxoaBPARdrQ)

Para resolver o problema acima, ajuda a multiplicar e dividir:

### Exemplo

var x = (0.2 \* 10 + 0.1 \* 10) / 10;       // x will be 0.3

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_inaccurate3&usg=ALkJrhibrhbPyWmvzs7olOINWfHUkeN8ag)

## Adicionando Números e Cordas

ATENÇÃO !!

O JavaScript usa o operador + para adição e concatenação.

Os números são adicionados. As cordas são concatenadas.

Se você adicionar dois números, o resultado será um número:

### Exemplo

var x = 10;  
var y = 20;  
var z = x + y;           // z will be 30 (a number)

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_add&usg=ALkJrhg25lGK-qmepVO7msUt1y2EKmX75g)

Se você adicionar duas strings, o resultado será uma concatenação de string:

### Exemplo

var x = "10";  
var y = "20";  
var z = x + y;           // z will be 1020 (a string)

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_add_strings1&usg=ALkJrhhj1g-KdT3KiN9Hl_6GixHeF1UCmg)

Se você adicionar um número e uma string, o resultado será uma concatenação de string:

### Exemplo

var x = 10;  
var y = "20";  
var z = x + y;           // z will be 1020 (a string)

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_add_strings2&usg=ALkJrhiVW0mkNJljmPXj062HexzMct51mA)

Se você adicionar uma string e um número, o resultado será uma concatenação de string:

### Exemplo

var x = "10";  
var y = 20;  
var z = x + y;           // z will be 1020 (a string)

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_add_strings5&usg=ALkJrhgtGwOq-I26eCCEN3CJTTIbIhyExg)

Um erro comum é esperar que este resultado seja 30:

### Exemplo

var x = 10;  
var y = 20;  
var z = "The result is: " + x + y;

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_add_strings3&usg=ALkJrhg29jyeWeFNHJEo8VKYGjhK66UVGA)

Um erro comum é esperar que esse resultado seja 102030:

### Exemplo

var x = 10;  
var y = 20;  
var z = "30";  
var result = x + y + z;

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_add_strings4&usg=ALkJrhgYvq5GqjH49alPaG2uxOBbrwSvRQ)

O compilador JavaScript funciona da esquerda para a direita.

Primeiro 10 + 20 é adicionado porque x e y são ambos os números.

Então 30 + "30" é concatenado porque z é uma string.

## Cordas Numéricas

Cadeias de caracteres JavaScript podem ter conteúdo numérico:

var x = 100;         // x is a number  
  
var y = "100";       // y is a string

O JavaScript tentará converter strings em números em todas as operações numéricas:

Isso funcionará:

var x = "100";  
var y = "10";  
var z = x / y;       // z will be 10

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_string1&usg=ALkJrhiAS8OrUQwnGWn6OqxlvPlh58YyNw)

Isso também funcionará:

var x = "100";  
var y = "10";  
var z = x \* y;       // z will be 1000

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_string2&usg=ALkJrhgCkPAVxzWaKygv3rpwDPHpfrbv-Q)

E isso vai funcionar:

var x = "100";  
var y = "10";  
var z = x - y;       // z will be 90

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_string3&usg=ALkJrhh8nR3_rrdKgdvqxEyDqhvDggdYHg)

Mas isso não vai funcionar:

var x = "100";  
var y = "10";  
var z = x + y;       // z will not be 110 (It will be 10010)

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_string4&usg=ALkJrhjzTT7xwjpQlT3zr9yy-veUDMAoBg)

No último exemplo, o JavaScript usa o operador + para concatenar as strings.

## NaN - Não é um número

NaN é uma palavra reservada JavaScript que indica que um número não é um número legal.

Tentando fazer aritmética com uma seqüência não-numérica resultará em NaN (Não é um Número):

### Exemplo

var x = 100 / "Apple";  // x will be NaN (Not a Number)

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_divide_string&usg=ALkJrhjS0loXpTAshMkqVi1P2UfyytbE5Q)

No entanto, se a string contiver um valor numérico, o resultado será um número:

### Exemplo

var x = 100 / "10";     // x will be 10

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_divide_number&usg=ALkJrhjRB2wsATVdd4PJCygKl0HUZycLDg)

Você pode usar a função JavaScript global isNaN () para descobrir se um valor é um número:

### Exemplo

var x = 100 / "Apple";  
isNaN(x);               // returns true because x is Not a Number

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_isnan_true&usg=ALkJrhi3frOauG-yWCZ-ZCK6aywpfyRL_Q)

Cuidado com a NaN. Se você usa NaN em uma operação matemática, o resultado também será NaN:

### Exemplo

var x = NaN;  
var y = 5;  
var z = x + y;         // z will be NaN

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_nan_math&usg=ALkJrhgfbxEMNNrfkibrG4QsFxzyatWFDQ)

Ou o resultado pode ser uma concatenação:

### Exemplo

var x = NaN;  
var y = "5";  
var z = x + y;         // z will be NaN5

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_nan_concat&usg=ALkJrhh1xm8CjGwghUSMogQIvxrHCRsIBQ)

NaN é um número: tipo de número de devolução NaN:

### Exemplo

typeof NaN;            // returns "number"

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_nan_typeof&usg=ALkJrhgH_iBNKxeTBaP3QbTZat1Y4-pdyQ)

## Infinidade

Infinity (ou -Infinity) é o valor que o JavaScript retornará se você calcular um número fora do maior número possível.

### Exemplo

var myNumber = 2;  
while (myNumber != Infinity) {          // Execute until Infinity  
    myNumber = myNumber \* myNumber;  
}

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_infinity&usg=ALkJrhgmk1w-M7zWVSUhhLhUhes9gfNkUg)

Divisão de 0 (zero) também gera Infinity:

### Exemplo

var x =  2 / 0;          // x will be Infinity  
var y = -2 / 0;          // y will be -Infinity

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_infinity_zero&usg=ALkJrhiR3zr2B-0rBY28GgqN-nmZveLAcQ)

Infinity é um número: typeof Infinity retorna o número.

### Exemplo

typeof Infinity;        // returns "number"

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_infinity_typeof&usg=ALkJrhhJwNyKu97y9mNa1rYBjCQfFPN5EA)

## Hexadecimal

O JavaScript interpreta as constantes numéricas como hexadecimais se forem precedidas de 0x.

### Exemplo

var x = 0xFF;           // x will be 255

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_hex&usg=ALkJrhguphwJ1MD-s897dhp7nY8aymyylA)

Nunca escreva um número com um zero inicial (como 07).   
Algumas versões de JavaScript interpretam os números como octal se eles estiverem escritos com um zero inicial.

Por padrão, o JavaScript exibe números como base de 10 decimais.

Mas você pode usar o método toString () para produzir números como base 16 (hexadecimal), base 8 (octal) ou base 2 (binário).

### Exemplo

var myNumber = 128;  
myNumber.toString(16);  // returns 80  
myNumber.toString(8);   // returns 200  
myNumber.toString(2);   // returns 10000000

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_numbers_tostring&usg=ALkJrhgfHnaLCDbUOFDP9Oyntgzza7w6JA)

## Números podem ser objetos

Normalmente, os números de JavaScript são valores primitivos criados a partir de literais:

**var x = 123;**

Mas os números também podem ser definidos como objetos com a palavra-chave nova:

**var y = novo número (123);**

### Exemplo

var x = 123;  
var y = new Number(123);  
  
// typeof x returns number  
// typeof y returns object

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_number_object&usg=ALkJrhhcKVsjyQ0yGWxXzc1MkxPn9hmdcQ)

Não crie objetos numéricos. Isso retarda a velocidade de execução.   
A **nova** palavra-chave complica o código. Isso pode produzir alguns resultados inesperados:

Ao usar o operador ==, os números iguais são iguais:

### Exemplo

var x = 500;               
var y = new Number(500);  
  
// (x == y) is true because x and y have equal values

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_number_object1&usg=ALkJrhgAL62uRRYFI42nmBa7_zkEF66waw)

Ao usar o operador ===, os números iguais não são iguais, porque o operador === espera igualdade em ambos os tipos e valores.

### Exemplo

var x = 500;               
var y = new Number(500);  
  
// (x === y) is false because x and y have different types

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_number_object2&usg=ALkJrhhG7YZoJSIunvUkYQX6qUCD2q5ckA)

Ou pior ainda. Os objetos não podem ser comparados:

### Exemplo

var x = new Number(500);               
var y = new Number(500);  
  
// (x == y) is false because objects cannot be compared

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_number_object3&usg=ALkJrhjcLAJdCPKjf3MumaKv69uAkULl7w)

Observe a diferença entre (x == y) e (x === y).   
A comparação de dois objetos JavaScript sempre retornará falsa.