

Conversão Numérica



A forma mais simples de converter um número é por meio da invocação da função construtora Number

A screenshot of a macOS terminal window titled "number_conversion_1.js — javascriptmasterclass". The window is split into two panes: a code editor on the left and a terminal on the right.

Code Editor (Left Pane):

```
JS number_conversion_1.js ×
1 Number("10");
2 Number("9.9");
3 Number("0xFF");
4 Number("0b10");
5 Number("0o10");
6 Number();
7 Number("JavaScript");
8
```

Terminal (Right Pane):

```
TERMINAL ⚙️ 1: node
rodrigobranas:javascriptmasterclass $ node
> Number("10");
10
> Number("9.9");
9.9
> Number("0xFF");
255
> Number("0b10");
2
> Number("0o10");
8
> Number();
0
> Number("JavaScript");
NaN
> █
```

Um outro jeito de realizar conversões é por meio dos **operadores numéricos**

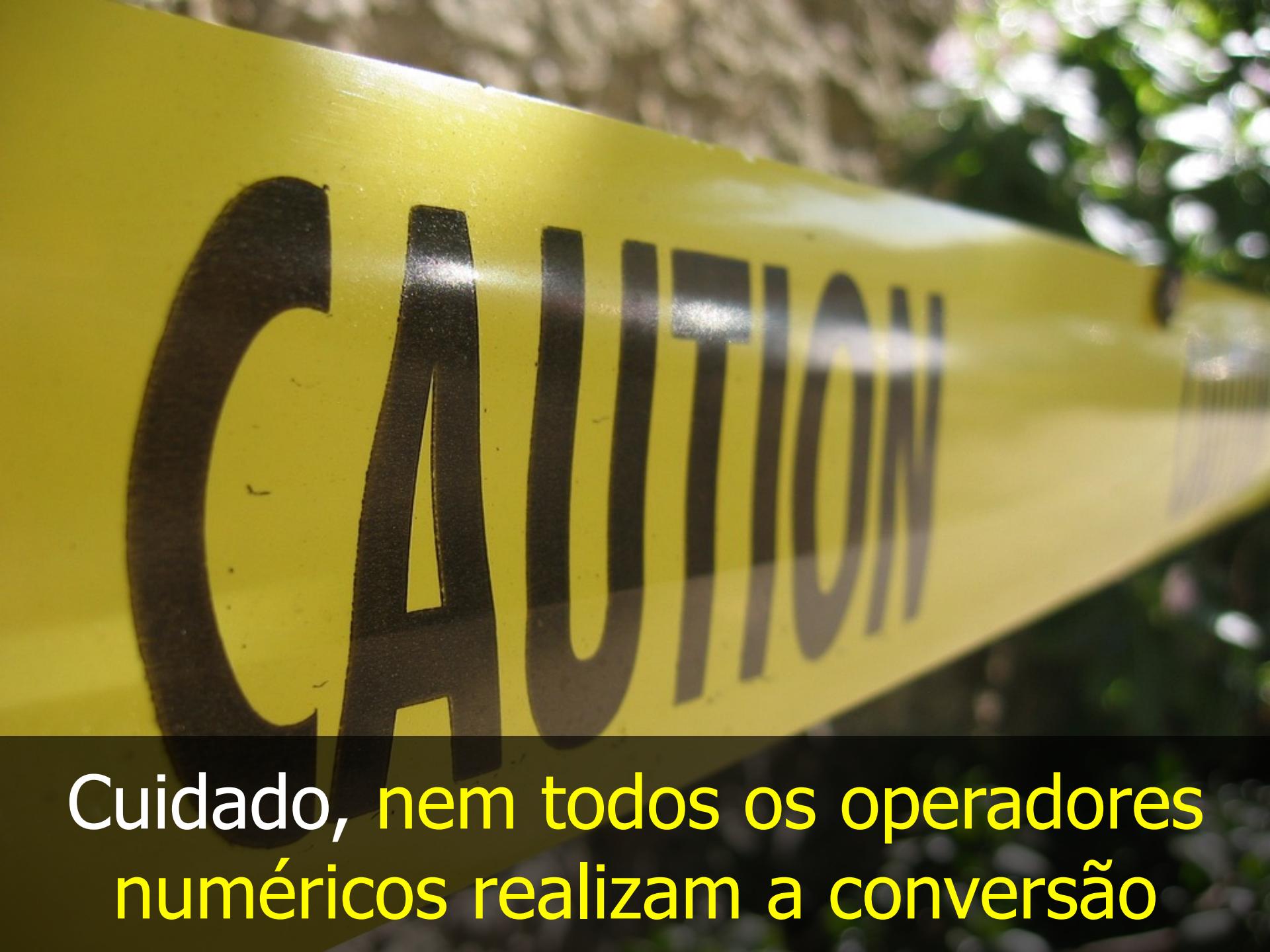
number_conversion_2.js — javascriptmasterclass

TERMINAL ... 1: node

```
rodrigobranas:javascriptmasterclass $ node
> ~~"10";
10
> +"10";
10
> "10" - 0;
10
> "10" * 1;
10
> "10" / 1;
10
> █
```

JS number_conversion_2.js x

```
1 ~"10";
2 +"10";
3 "10" - 0;
4 "10" * 1;
5 "10" / 1;
6
```

A close-up photograph of a yellow caution tape. The word "CAUTION" is printed in large, bold, black capital letters. The tape is slightly curved, and the background shows some green foliage.

CAUTION

Cuidado, nem todos os operadores
numéricos realizam a conversão

A screenshot of a Mac OS X desktop environment showing a terminal window. The window title is "number_conversion_3.js — javascriptmasterclass". The terminal pane displays the command "node" followed by the script content and its output.

The script content in the editor pane is:

```
JS number_conversion_3.js ×
1 "10" + 0;
2
```

The terminal output is:

```
TERMINAL ⌂ ⌓ ⌚ ...
rodrigobranas:javascriptmasterclass $ node
> "10" + 0;
'100'
>
```

O método `toString` de um número,
permite **convertê-lo para qualquer sistema
de numeração**, bastando indicar qual
é a base desejada

number_conversion_4.js — javascriptmasterclass

JS number_conversion_4.js x

TERMINAL ... 1: node

```
rodrigobranas:javascriptmasterclass $ node
> (0xA).toString(10);
'10'
> (0b1010).toString(16);
'a'
> (010).toString(2);
'1000'
> (10).toString(8);
'12'
> █
```

The screenshot shows a terminal window with the following content:

```
rodrigobranas:javascriptmasterclass $ node
> (0xA).toString(10);
'10'
> (0b1010).toString(16);
'a'
> (010).toString(2);
'1000'
> (10).toString(8);
'12'
> █
```

O método parseInt permite converter uma String para um número. Para isso basta indicar o **número e a sua base**, que caso não seja informada será 10.

number_conversion_5.js — javascriptmasterclass

JS number_conversion_5.js x

TERMINAL ... 1: node

```
rodrigobranas:javascriptmasterclass $ node
> parseInt("10", 10);
10
> parseInt("9.9", 10);
9
> parseInt("A", 16);
10
> parseInt("11", 2);
3
> parseInt("010", 8);
8
> █
```

1 parseInt("10", 10);
2 parseInt("9.9", 10);
3 parseInt("A", 16);
4 parseInt("11", 2);
5 parseInt("010", 8);
6

O método `parseFloat` é um pouco mais específico e converte **apenas** números no sistema de numeração decimal

number_conversion_6.js — javascriptmasterclass

JS number_conversion_6.js x

1 parseFloat("10");
2 parseFloat("2.5");
3 parseFloat("0xFF");
4 parseFloat("0b10");
5

TERMINAL ... 1: node

rodrigobranas:javascriptmasterclass \$ node
> parseFloat("10");
10
> parseFloat("2.5");
2.5
> parseFloat("0xFF");
0
> parseFloat("0b10");
0
> █