



Number

O tipo Number é primitivo, imutável e é representado internamente pelo padrão **IEEE 754** de 64 bits



Suporta os sistemas de numeração
decimal, hexadecimal, binário e octal

O sistema de numeração **decimal**, de **base 10**, deve iniciar com um número de 1 a 9, seguido por números de 0 a 9 com ou sem ponto, indicando se é inteiro ou decimal

Decimal	Hexadecimal	Binário	Octal
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			
31			
32			

O sistema de numeração **hexadecimal**, de base **16**, deve iniciar com 0x ou 0X, seguido por números de 0 a 9 e letras de A a F

Decimal	Hexadecimal	Binário	Octal
0	0		
1	1		
2	2		
3	3		
4	4		
5	5		
6	6		
7	7		
8	8		
9	9		
10	A		
11	B		
12	C		
13	D		
14	E		
15	F		
16	10		
17	11		
18	12		
19	13		
20	14		
21	15		
22	16		
23	17		
24	18		
25	19		
26	1A		
27	1B		
28	1C		
29	1D		
30	1E		
31	1F		
32	20		

O sistema de numeração **binário**, de **base 2**, deve iniciar com 0b ou 0B, seguido por números de 0 a 1

Decimal	Hexadecimal	Binário	Octal
0	0	000000	
1	1	000001	
2	2	000010	
3	3	000011	
4	4	000100	
5	5	000101	
6	6	000110	
7	7	000111	
8	8	001000	
9	9	001001	
10	A	001010	
11	B	001011	
12	C	001100	
13	D	001101	
14	E	001110	
15	F	001111	
16	10	010000	
17	11	010001	
18	12	010010	
19	13	010011	
20	14	010100	
21	15	010101	
22	16	010110	
23	17	010111	
24	18	011000	
25	19	011001	
26	1A	011010	
27	1B	011011	
28	1C	011100	
29	1D	011101	
30	1E	011110	
31	1F	011111	
32	20	100000	

O sistema de numeração **octal**, de **base 8**,
deve iniciar com 0, 0o ou 0O seguido por
números de 0 a 7

Decimal	Hexadecimal	Binário	Octal
0	0	000000	0
1	1	000001	1
2	2	000010	2
3	3	000011	3
4	4	000100	4
5	5	000101	5
6	6	000110	6
7	7	000111	7
8	8	001000	10
9	9	001001	11
10	A	001010	12
11	B	001011	13
12	C	001100	14
13	D	001101	15
14	E	001110	16
15	F	001111	17
16	10	010000	20
17	11	010001	21
18	12	010010	22
19	13	010011	23
20	14	010100	24
21	15	010101	25
22	16	010110	26
23	17	010111	27
24	18	011000	30
25	19	011001	31
26	1A	011010	32
27	1B	011011	33
28	1C	011100	34
29	1D	011101	35
30	1E	011110	36
31	1F	011111	37
32	20	100000	40

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor. The terminal window is titled 'number_1.js — javascriptmasterclass' and has a tab bar with 'TERMINAL' and '1: node'. The terminal content shows the output of running the script 'number_1.js' with various numeric literals.

The code editor window contains the file 'number_1.js' with the following content:

```
1 10;
2 9.9;
3 0xFF;
4 0b10;
5 0o10;
```

The terminal output is as follows:

```
rodrigobranas:javascriptmasterclass $ node
> 10;
10
> 9.9;
9.9
> 0xFF;
255
> 0b10;
2
> 0o10;
8
> █
```

A screenshot of a Mac OS X desktop environment showing a terminal window. The window title is "number_2.js — javascriptmasterclass". The terminal pane displays the output of running the file "node number_2.js".

The code in "number_2.js" is as follows:

```
1 new Number(10);
2 new Number(9.9);
3 new Number(0xFF);
4 new Number(0b10);
5 new Number(0o10);
6
```

The terminal output shows the results of each new Number() call:

```
rodrigobranas:javascriptmasterclass $ node
> new Number(10);
[Number: 10]
> new Number(9.9);
[Number: 9.9]
> new Number(0xFF);
[Number: 255]
> new Number(0b10);
[Number: 2]
> new Number(0o10);
[Number: 8]
>
```

Os métodos `toExponential`, `toFixed` e `toPrecision` podem ser utilizados para mudar a forma como um número é representado

number_3.js — javascriptmasterclass

JS number_3.js ×

```
1 (123.4).toExponential(10);
2 (123.4).toFixed(10);
3 (123.4).toPrecision(10);
4
```

TERMINAL ... 1: node

```
> (123.4).toExponential(10);
'1.234000000e+2'
> (123.4).toFixed(10);
'123.400000000'
> (123.4).toPrecision(10);
'123.400000'
```