

Number

O tipo number é primitivo e baseado no padrão **IEEE 754** com 64 bits

1. `10;` // 10
2. `9.9;` // 9.9
3. `0xFF;` // 255 (utiliza números de 0 a 9 e letras de A a F)
4. `0b10;` // 2 (utiliza somente 0 e 1)
5. `0o10;` // 8 (utiliza somente números de 0 a 7)

Os tipos de number são: **inteiro, ponto flutuante, hexadecimal, binário e octal**

1. `0.1 + 0.2; // 0.30000000000000004`
2. `1/0; // Infinity`
3. `(1 + 2 + 3 + 4 + 5 + 6) * undefined; // NaN`

Existem algumas **exceções**

1. `Math.random(); // 0.978686876...`
2. `Math.pow(2,4); // 16`
3. `Math.floor(-9.9); // -10`
4. `Math.ceil(-9.9); // -9`
5. `Math.trunc(-9.9); // -9`
6. `Math.round(-9.9); // -10`

Math API



String

O tipo string é primitivo, imutável e baseado em **unicode**, com 16 bits, codificado em UTF-16

1. `'JavaScript';`
2. `"JavaScript";`
3. ``JavaScript`;`

É possível declarar uma string utilizando
" / "" ou ` `


```
1. let multilineString = "Curabitur blandit tempus" +  
2. "porttitor. Nullam quis risus eget urna mollis " +  
3. "ornare vel eu leo. Vestibulum id ligula porta " +  
4. "felis euismod semper. Fusce dapibus, tellus ac " +  
5. "cursus commodo, tortor mauris condimentum " +  
6. "nibh, ut fermentum massa justo sit amet risus."
```

Multiline utilizando o operador +

```
1. let multilineString = "Curabitur blandit tempus \
2. porttitor. Nullam quis risus eget urna mollis \
3. ornare vel eu leo. Vestibulum id ligula porta \
4. felis euismod semper. Fusce dapibus, tellus ac \
5. cursus commodo, tortor mauris condimentum \
6. nibh, ut fermentum massa justo sit amet risus."
```

Multiline utilizando \

\0 the NULL character
\' single quote
\" double quote
\\ backslash
\n new line
\r carriage return
\v vertical tab
\t tab
\b backspace
\f form feed
\uXXXXX unicode codepoint
\u{X} ... \u{XXXXXX} unicode codepoint
\xXX the Latin-1 character

Escape Characters

1. `let language = "JavaScript";`
2. `let interpolateString = `${language} is one of the most popular languages in the world`; // 'JavaScript is one of the most popular languages in the world'`

Interpolando valores com Template Literals

1. `"á" > "b" // true`
2. `"á".charCodeAt() > "b".charCodeAt() // true`
3. `255 > 98 // true`

Comparação de String

```
1.  typeof "JavaScript"; // string
2.  typeof (new String("JavaScript")); // object
3.  typeof (new String("JavaScript")).valueOf(); // string
4.  "JavaScript".charCodeAt(1); // 61
5.  "JavaScript".replace("a", 4); // 'J4vaScript'
6.  "JavaScript".replace(/a/g, 4); // 'J4v4Script'
7.  "Java Script".split(" "); // ['Java', 'Script']
8.  "á".localeCompare("b"); // -1
9.  "JavaScript".repeat(2); // 'JavaScriptJavaScript'
10. "JavaScript".startsWith("Java"); // true
11. "  Java Script  ".trim(); // 'Java Script'
12. "á".localeCompare("b"); // -1
```

String API



Boolean

O tipo boolean é primitivo, imutável e representado pelos valores **true** e **false**

1. `!!0; // false`
2. `!!"; // false`
3. `!!undefined; // false`
4. `!!null; // false`
5. `!!NaN; // false`
6. `!!false; // false`
7. `!!{}; // true`
8. `!![]; // true`

Utilizando a operação ToBoolean para analisar o comportamento booleano

Cuidado com a **coersão de tipo** em
comparações utilizando os operadores
== e !=

1. `0 == ""; // true`
2. `0 == '0'; // true`
3. `false == undefined; // false`
4. `false == null; // false`
5. `null == undefined; // true`
6. `1 == true; // true`
7. `0 == false; // true`
8. `0 == '\n'; // true`

O resultado da coerção de tipo em muitos casos pode ser **inesperado**

Prefira sempre a utilização dos
operadores **===** e **!==**

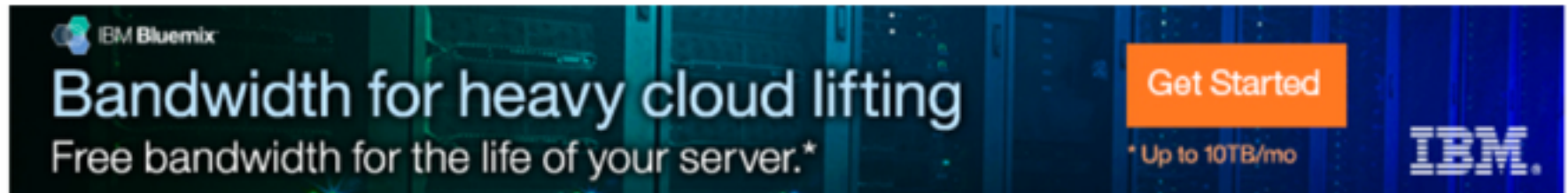
1. `0 || 2; // 2`
2. `1 || 2; // 1`
3. `1 && 2; // 2`
4. `0 && 2; // 0`

Os operadores lógicos `||` e `&&` podem ser utilizados de diversas formas



Symbol

Why bring symbols to javascript?



IBM Bluemix

Bandwidth for heavy cloud lifting

Free bandwidth for the life of your server.*

Get Started

* Up to 10TB/mo

IBM.



151



27

As you may know they are [planning to include](#) new Symbol primitive type in ECMAScript 6 (not to mention some other crazy stuff). I always thought that the `:symbol` notion in Ruby is needless; we could easily use plain strings instead, like we do in JavaScript. And now they decide to complicate things in JS with that.

I don't understand the motivation. Could someone explain to me whether we really need symbols in JavaScript?

UPDATE: Recently a [brilliant article from Mozilla](#) came up. Read it if you're curious.

javascript

symbols

ecmascript-harmony

ecmascript-6

share edit flag

edited Jun 16 '15 at 8:23

asked Feb 12 '14 at 9:53



Yanis T

977 ● 2 ● 9 ● 12

O tipo symbol é primitivo, único e imutável e serve para **identificar uma propriedade de um objeto** de forma única, promovendo privacidade e prevenindo a sobrescrita, ao contrário da string



undefined

O tipo **undefined** é retornado caso uma propriedade de um determinado objeto seja consultada e não exista



null

O tipo **null** indica a ausência de valor em uma determinada propriedade já existente