



# Object

Um objeto é uma **coleção dinâmica** de **propriedades** definidas por chaves, que podem ser do tipo String ou Symbol, e valores que podem ser de qualquer tipo de dado

<b>Chave</b>	<b>Valor</b>
title	Clean Code
author	Robert C. Martin
pages	464
language	English
available	true

É possível criar objetos de várias formas: pela notação literal, por meio de uma função construtora ou do método `create` da Object API

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor.

The terminal window, titled "TERMINAL", shows the command "node" being run, followed by several empty object literals ({}). The terminal window has a title bar with "1: node" and a dropdown menu.

The code editor window, titled "object\_1.js — javascriptmasterclass", contains the following JavaScript code:

```
1  {}
2  new Object();
3  Object.create(null);
4
```



Existe diferença entre essas formas?

Uma das diversas maneiras de atribuir propriedades a um objeto é durante a sua inicialização, pela **notação literal**

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor side-by-side.

The terminal window on the right shows the command `node object/object_2.js` being run, followed by the output of the script. The output is a JSON object representing a book:

```
rodrigobranas:javascriptmasterclass $ node object/object_2.js
{ title: 'Clean Code',
  author: 'Robert C. Martin',
  pages: 464,
  language: 'English',
  available: true }
rodrigobranas:javascriptmasterclass $
```

The code editor on the left contains the script `object_2.js`:

```
JS object_2.js x
object_2.js — javascriptmasterclass

1 const book = {
2   title: "Clean Code",
3   author: "Robert C. Martin",
4   pages: 464,
5   language: "English",
6   available: true
7 };
8 console.log(book);
9
```

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor.

The terminal window is titled "object\_3.js — javascriptmasterclass". It contains the command:

```
rodrigobranas:javascriptmasterclass $ node object/object_3.js
```

The code editor window is titled "JS object\_3.js". It displays the following JavaScript code:

```
1 const title = "Clean Code";
2 const author = "Robert C. Martin";
3 const pages = 464;
4 const language = "English";
5 const available = true;
6 const book = {
7   title,
8   author,
9   pages,
10  language,
11  available
12 };
13 console.log(book);
14
```

The terminal output shows the execution of the script, resulting in the following object:

```
{ title: 'Clean Code',
  author: 'Robert C. Martin',
  pages: 464,
  language: 'English',
  available: true }
```

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

Dependendo da chave é necessário  
declará-la diretamente como String

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor side-by-side.

The terminal window on the right has a title bar "TERMINAL" and a dropdown menu "1: bash". The command run is "node object/object\_4.js". The output is:

```
rodrigobranas:javascriptmasterclass $ node object/object_4.js
{ title: 'Clean Code',
  author: 'Robert C. Martin',
  'number-of-pages': 464,
  language: 'English',
  available: true }
```

The code editor on the left has a title bar "object\_4.js — javascriptmasterclass". It contains the following JavaScript code:

```
1 const book = {
2   title: "Clean Code",
3   author: "Robert C. Martin",
4   "number-of-pages": 464,
5   language: "English",
6   available: true
7 };
8 console.log(book);
9
```

Também é possível computar as chaves **em tempo de execução**

A screenshot of a terminal window titled "object\_5.js — javascriptmasterclass". The terminal shows the command "node object/object\_5.js" followed by the output of a console.log statement. The output is an object with five properties: title, author, pages, language, and available, all set to specific values.

```
JS object_5.js x
object_5.js — javascriptmasterclass
1 const key1 = "title";
2 const key2 = "author";
3 const key3 = "pages";
4 const key4 = "language";
5 const key5 = "available";
6 const book = {
7   [key1]: "Clean Code",
8   [key2]: "Robert C. Martin",
9   [key3]: 464,
10  [key4]: "English",
11  [key5]: true
12 };
13 console.log(book);
14

TERMINAL ... 1: bash + = 1
rodrigobranas:javascriptmasterclass $ node object/object_5.js
{ title: 'Clean Code',
  author: 'Robert C. Martin',
  pages: 464,
  language: 'English',
  available: true }
rodrigobranas:javascriptmasterclass $
```

Além da notação literal, é possível atribuir propriedades aos objetos por meio da sua referência

A screenshot of a Mac OS X desktop environment showing a terminal window. The window title is "object\_6.js — javascriptmasterclass". The terminal tab is labeled "1: bash". The command run is "node object/object\_6.js". The output shows the execution of a JavaScript file that creates an object representing a book and logs it to the console.

```
JS object_6.js x
object_6.js — javascriptmasterclass
TERMINAL ... 1: bash + = 1
rodrigobranas:javascriptmasterclass $ node object/object_6.js
{ title: 'Clean Code',
  author: 'Robert C. Martin',
  pages: 464,
  language: 'English',
  available: true }
rodrigobranas:javascriptmasterclass $
```

```
1 const book = {};
2 book.title = "Clean Code";
3 book.author = "Robert C. Martin";
4 book.pages = 464;
5 book.language = "English";
6 book.available = true;
7 console.log(book);
8
```

Assim como na notação literal, é possível computar as chaves de um objeto **em tempo de execução** por meio da sua referência

A screenshot of a Mac OS X desktop environment showing a terminal window. The window title is "object\_7.js — javascriptmasterclass". The main pane displays the contents of a JavaScript file named "object\_7.js". The code defines a book object with properties: title, author, pages, language, and available status. The file ends with a call to console.log(book). The right-hand sidebar contains a "TERMINAL" section with a dropdown menu set to "1: bash". Below the dropdown, the terminal output shows the execution of the script using node, followed by the printed object structure, and ends with a command prompt.

```
JS object_7.js x
object_7.js — javascriptmasterclass
1 const key1 = "title";
2 const key2 = "author";
3 const key3 = "pages";
4 const key4 = "language";
5 const key5 = "available";
6 const book = {};
7 book[key1] = "Clean Code",
8 book[key2] = "Robert C. Martin",
9 book[key3] = 464,
10 book[key4] = "English",
11 book[key5] = true
12 console.log(book);
13

TERMINAL ... 1: bash +
rodrigobranas:javascriptmasterclass $ node object/object_7.js
{ title: 'Clean Code',
  author: 'Robert C. Martin',
  pages: 464,
  language: 'English',
  available: true }
rodrigobranas:javascriptmasterclass $
```

Cada uma das propriedades de um  
objeto podem ser consultadas por meio  
da sua referência, **de forma direta**

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor side-by-side.

The terminal window on the right shows the command `node object/object_8.js` being run, followed by the output:

```
rodrigobranas:javascriptmasterclass $ node object/object_8.js
Clean Code
Robert C. Martin
464
English
true
rodrigobranas:javascriptmasterclass $
```

The code editor on the left contains the file `object_8.js`:

```
JS object_8.js ×
object_8.js — javascriptmasterclass
1 const book = {
2   title: "Clean Code",
3   author: "Robert C. Martin",
4   pages: 464,
5   language: "English",
6   available: true
7 };
8 console.log(book.title);
9 console.log(book.author);
10 console.log(book.pages);
11 console.log(book.language);
12 console.log(book.available);
13
```

É possível consultar cada uma das propriedades de um objeto por meio da computação das chaves

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor.

The terminal window (top right) has the title "TERMINAL" and shows the command "node object/object\_9.js" followed by the output:

```
rodrigobranas:javascriptmasterclass $ node object/object_9.js
Clean Code
Robert C. Martin
464
English
true
rodrigobranas:javascriptmasterclass $
```

The code editor window (left side) has the title "object\_9.js — javascriptmasterclass". It contains the following JavaScript code:

```
1 const book = {
2     title: "Clean Code",
3     author: "Robert C. Martin",
4     pages: 464,
5     language: "English",
6     available: true
7 };
8 for (let key in book) {
9     console.log(book[key]);
10}
11
```

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor side-by-side.

The terminal window on the right shows the command `node object/object_10.js` being run, followed by the output of the script's console.log statement:

```
rodrigobranas:javascriptmasterclass $ node object/object_10.js
{ title: 'Clean Code',
  author: 'Robert C. Martin',
  pages: 464,
  language: 'English',
  available: true }
rodrigobranas:javascriptmasterclass $
```

The code editor on the left contains the script `object_10.js`:

```
JS object_10.js ×
object_10.js — javascriptmasterclass

1 const book1 = {
2   title: "Clean Code",
3   author: "Robert C. Martin",
4   pages: 464,
5   language: "English",
6   available: true
7 };
8 const book2 = {};
9 for (let key in book1) {
10   book2[key] = book1[key];
11 }
12 console.log(book2);
13
```