

A blurry, out-of-focus photograph of a row of filing cabinets. The cabinets are light-colored with dark horizontal stripes across them. Some have small circular labels or badges. The perspective is from a low angle, looking up at the top of the row.

# Object API

O método **Object.assign** faz a cópia das propriedades dos objetos passados por parâmetro para o objeto alvo, que é retornado

A screenshot of a macOS desktop environment showing a terminal window. The window title is "object\_api\_1.js — javascriptmasterclass". The terminal pane displays the output of a Node.js command:

```
rodrigobranas:javascriptmasterclass $ node object_api/object_api_1.js
{ name: 'JavaScript', year: 1995, paradigm: 'OO and Functional' }
rodrigobranas:javascriptmasterclass $
```

The left side of the image shows the code editor pane with the file "object\_api\_1.js" open. The code is as follows:

```
1 const javascript = Object.create({});  
2 Object.assign(javascript, {  
3     name: "JavaScript",  
4     year: 1995,  
5     paradigm: "OO and Functional"  
6 });  
7 console.log(javascript);  
8
```

A screenshot of a Mac OS X desktop environment showing a terminal window. The window title is "object\_api\_2.js — javascriptmasterclass". The main pane displays the contents of the file "object\_api\_2.js". The code creates a new object "javascript" using `Object.create({})` and then uses `Object.assign` to add properties: "name" (value "JavaScript"), "year" (value "1995"), "paradigm" (value "OO and Functional"), "author" (value "Brendan Eich"), and "influencedBy" (value "Java, Scheme and Self"). Finally, it logs the object to the console with `console.log(javascript);`. The right-hand terminal pane shows the output of running the script with "node object\_api/object\_api\_2.js", which outputs the object with its properties and values.

```
JS object_api_2.js x
object_api_2.js — javascriptmasterclass
1 const javascript = Object.create({});
2 Object.assign(javascript, {
3   name: "JavaScript",
4   year: 1995,
5   paradigm: "OO and Functional"
6 }, {
7   author: "Brendan Eich",
8   influencedBy: "Java, Scheme and Self"
9 });
10 console.log(javascript);
11
```

```
TERMINAL ... 1: bash
rodrigobranas:javascriptmasterclass $ node object_api/object_api_2.js
{ name: 'JavaScript',
  year: 1995,
  paradigm: 'OO and Functional',
  author: 'Brendan Eich',
  influencedBy: 'Java, Scheme and Self' }
rodrigobranas:javascriptmasterclass $
```

O método **Object.keys** retorna as chaves das propriedades do objeto

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_api/object_api_3.js` being run, followed by the output `[ 'name', 'year', 'paradigm' ]`.

The code editor on the left contains the following JavaScript code:

```
JS object_api_3.js x
object_api_3.js — javascriptmasterclass

1 const javascript = {
2     name: "JavaScript",
3     year: 1995,
4     paradigm: "OO and Functional"
5 };
6 console.log(Object.keys(javascript));
7
```

O método `Object.values` retorna os valores das propriedades do objeto

A screenshot of a Mac OS X desktop environment. On the left is a code editor window titled "object\_api\_4.js — javascriptmasterclass". The file contains the following JavaScript code:

```
JS object_api_4.js x
1 const javascript = {
2     name: "JavaScript",
3     year: 1995,
4     paradigm: "OO and Functional"
5 };
6 console.log(Object.values(javascript));
7
```

The code defines an object named "javascript" with properties: name, year, and paradigm. It then logs the values of this object to the console using Object.values(). On the right side of the screen is a terminal window titled "TERMINAL" with the identifier "1: bash". The terminal shows the command "node object\_api/object\_api\_4.js" being run, followed by the output "[ 'JavaScript', 1995, 'OO and Functional' ]".

O método `Object.entries` retorna as propriedades do objeto em pares de chave e valor

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

The terminal window on the right shows the command `node object_api/object_api_5.js` being run, followed by the output:

```
rodrigobranas:javascriptmasterclass $ node object_api/object_api_5.js
[ [ 'name', 'JavaScript' ],
  [ 'year', 1995 ],
  [ 'paradigm', 'OO and Functional' ] ]
rodrigobranas:javascriptmasterclass $
```

The code editor on the left contains the file `object_api_5.js`:

```
JS object_api_5.js ×
object_api_5.js — javascriptmasterclass

1  const javascript = {
2      name: "JavaScript",
3      year: 1995,
4      paradigm: "OO and Functional"
5  };
6  console.log(Object.entries(javascript));
7
```

O método `Object.is` compara dois objetos, considerando os tipos de dados, de forma similar ao operador `====`

A screenshot of a Mac OS X desktop environment. On the left is a code editor window titled "object\_api\_6.js — javascriptmasterclass". The file contains the following JavaScript code:

```
JS object_api_6.js x
object_api_6.js — javascriptmasterclass
1 const javascript = {
2   name: "JavaScript",
3   year: 1995,
4   paradigm: "OO and Functional"
5 };
6 console.log(Object.is(javascript, javascript));
7
```

The code defines an object named "javascript" with properties: name ("JavaScript"), year (1995), and paradigm ("OO and Functional"). It then logs the result of `Object.is(javascript, javascript)` to the console.

To the right of the code editor is a terminal window titled "TERMINAL" with a tab labeled "1: bash". The terminal shows the command `node object\_api/object\_api\_6.js` being run, followed by the output "true".

A screenshot of a Mac OS X desktop environment showing a terminal window. The window title is "object\_api\_7.js — javascriptmasterclass". The main pane contains the following JavaScript code:

```
1 const javascript1 = {  
2     name: "JavaScript",  
3     year: 1995,  
4     paradigm: "OO and Functional"  
5 };  
6 const javascript2 = {  
7     name: "JavaScript",  
8     year: 1995,  
9     paradigm: "OO and Functional"  
10};  
11 console.log(Object.is(javascript1, javascript2));  
12
```

The code defines two objects, javascript1 and javascript2, which are identical except for their numerical year values. The final line uses the `Object.is` method to compare these two objects.

To the right of the code editor is a terminal pane titled "TERMINAL" with a dropdown menu showing "1: bash". The terminal output shows the command being run and the resulting output:

```
rodrigobranas:javascriptmasterclass $ node object_api/object_api_7.js  
false  
rodrigobranas:javascriptmasterclass $
```

A screenshot of a macOS terminal window titled "object\_api\_8.js — javascriptmasterclass". The window has three panes: a code editor on the left, a terminal on the right, and a status bar at the bottom.

The code editor pane shows the file "object\_api\_8.js" with the following content:

```
1 console.log(Object.is(NaN, NaN));
```

The terminal pane shows the output of running the script:

```
rodrigobranas:javascriptmasterclass $ node object_api/object_api_8.js
true
rodrigobranas:javascriptmasterclass $
```

# defineProperty

configurable – Permite que uma determinada propriedade seja apagada

enumerable – Permite que uma determinada propriedade seja enumerada

value – Define o valor de uma determinada propriedade

writable – Permite que uma determinada propriedade tenha seu valor modificado

A screenshot of a macOS desktop environment showing a terminal window and a code editor. The terminal window is titled 'object\_api\_9.js — javascriptmasterclass' and contains the command 'node object\_api/object\_api\_9.js'. The code editor shows the file 'object\_api\_9.js' with the following content:

```
1 const javascript = {};
2 Object.defineProperty(javascript, "name", {
3   value: "JavaScript"
4 });
5 console.log(javascript);
6 console.log(Object.keys(javascript));
7 console.log(Object.values(javascript));
8 console.log(Object.entries(javascript));
9
```

The terminal output shows the execution of the script, with the final command being 'rodrigobranas:javascriptmasterclass \$'.

A screenshot of a Mac OS X terminal window titled "object\_api\_10.js — javascriptmasterclass". The window is split into two panes: a code editor on the left and a terminal on the right.

The code editor pane shows the file "object\_api\_10.js" with the following content:

```
1 const javascript = {};
2 Object.defineProperty(javascript, "name", {
3     enumerable: true,
4     value: "JavaScript"
5 });
6 console.log(javascript);
7 console.log(Object.keys(javascript));
8 console.log(Object.values(javascript));
9 console.log(Object.entries(javascript));
10
```

The terminal pane shows the output of running the script:

```
rodrigobranas:javascriptmasterclass $ node object_api/object_api_10.js
{ name: 'JavaScript' }
[ 'name' ]
[ 'JavaScript' ]
[ [ 'name', 'JavaScript' ] ]
rodrigobranas:javascriptmasterclass $
```

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

The terminal window on the right shows the output of running the script:

```
rodrigobranas:javascriptmasterclass $ node object_api/object_api_11.js
{ name: 'JavaScript' }
[ 'name' ]
[ 'JavaScript' ]
[ [ 'name', 'JavaScript' ] ]
rodrigobranas:javascriptmasterclass $
```

The code editor on the left contains the following JavaScript code:

```
1 const javascript = {};
2 Object.defineProperty(javascript, "name", {
3     enumerable: true,
4     value: "JavaScript"
5 });
6 javascript.name = "ECMAScript";
7 console.log(javascript);
8 console.log(Object.keys(javascript));
9 console.log(Object.values(javascript));
10 console.log(Object.entries(javascript));
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_api/object_api_12.js` being run, followed by the output:

```
rodrigobranas:javascriptmasterclass $ node object_api/object_api_12.js
{ name: 'ECMAScript' }
[ 'name' ]
[ 'ECMAScript' ]
[ [ 'name', 'ECMAScript' ] ]
rodrigobranas:javascriptmasterclass $
```

The code editor on the left contains the file `object_api_12.js`:

```
1 const javascript = {};
2 Object.defineProperty(javascript, "name", {
3     enumerable: true,
4     value: "JavaScript",
5     writable: true
6 });
7 javascript.name = "ECMAScript";
8 console.log(javascript);
9 console.log(Object.keys(javascript));
10 console.log(Object.values(javascript));
11 console.log(Object.entries(javascript));
12
```

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor. The terminal window is titled 'object\_api\_13.js — javascriptmasterclass' and shows the command 'node object\_api/object\_api\_13.js' followed by its output: an object with a 'name' property set to 'ECMAScript', and arrays for keys, values, and entries.

The code editor window has tabs for 'object\_api\_13.js' and 'index.js'. The code in 'object\_api\_13.js' is as follows:

```
1 const javascript = {};
2 Object.defineProperty(javascript, "name", {
3     enumerable: true,
4     value: "JavaScript",
5     writable: true
6 });
7 javascript.name = "ECMAScript";
8 delete javascript.name;
9 console.log(javascript);
10 console.log(Object.keys(javascript));
11 console.log(Object.values(javascript));
12 console.log(Object.entries(javascript));
13
```

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

The terminal window on the right shows the command `node object_api/object_api_14.js` being run, followed by four empty square brackets indicating an array or object structure.

The code editor on the left contains the following JavaScript code:

```
1 const javascript = {};
2 Object.defineProperty(javascript, "name", {
3     configurable: true,
4     enumerable: true,
5     value: "JavaScript",
6     writable: true
7 });
8 javascript.name = "ECMAScript";
9 delete javascript.name;
10 console.log(javascript);
11 console.log(Object.keys(javascript));
12 console.log(Object.values(javascript));
13 console.log(Object.entries(javascript));
14
```

# preventExtensions, seal e freeze

preventExtensions – Impede que o objeto tenha novas propriedades, mas permite modificar ou remover as propriedades existentes

seal – Impede que o objeto tenha novas propriedades ou apague propriedades existentes, mas permite modificar propriedades existentes

freeze – Impede que o objeto tenha novas propriedades, apague ou modifique propriedades existentes

|                          | Create | Read | Update | Delete |
|--------------------------|--------|------|--------|--------|
| Object.preventExtensions | no     | yes  | yes    | yes    |
| Object.seal              | no     | yes  | yes    | no     |
| Object.freeze            | no     | yes  | no     | no     |

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 output of running `node object_api/object_api_15.js`. The output is:

```
rodrigobranas:javascriptmasterclass $ node object_api/object_api_15.js
{ name: 'ECMAScript', paradigm: 'OO and Functional' }
false
rodrigobranas:javascriptmasterclass $
```

The code editor window on the left contains the file `object_api_15.js`:

```
1 const javascript = {
2   name: "JavaScript",
3   year: 1995,
4   paradigm: "OO and Functional"
5 };
6 Object.preventExtensions(javascript);
7 javascript.name = "ECMAScript";
8 javascript.author = "Brendan Eich";
9 delete javascript.year;
10 console.log(javascript);
11 console.log(Object.isExtensible(javascript));
12
```

A screenshot of a Mac OS X desktop environment showing a terminal window. The window title is "object\_api\_16.js — javascriptmasterclass". The main pane contains the following JavaScript code:

```
1 const javascript = {
2     name: "JavaScript",
3     year: 1995,
4     paradigm: "OO and Functional"
5 };
6 Object.seal(javascript);
7 javascript.name = "ECMAScript";
8 javascript.author = "Brendan Eich";
9 delete javascript.year;
10 console.log(javascript);
11 console.log(Object.isExtensible(javascript));
12 console.log(Object.isSealed(javascript));
13
```

The right pane shows the output of running the script with node:

```
rodrigobranas:javascriptmasterclass $ node object_api/object_api_16.js
{ name: 'ECMAScript', year: 1995, paradigm: 'OO and Functional' }
false
true
rodrigobranas:javascriptmasterclass $
```

A screenshot of a macOS desktop environment showing a terminal window. The window title is "object\_api\_17.js — javascriptmasterclass". The main pane contains the following JavaScript code:

```
1 const javascript = {
2     name: "JavaScript",
3     year: 1995,
4     paradigm: "OO and Functional"
5 };
6 Object.freeze(javascript);
7 javascript.name = "ECMAScript";
8 javascript.author = "Brendan Eich";
9 delete javascript.year;
10 console.log(javascript);
11 console.log(Object.isExtensible(javascript));
12 console.log(Object.isSealed(javascript));
13 console.log(Object.isFrozen(javascript));
14
```

The right pane shows the output of running the script with node:

```
rodrigobranas:javascriptmasterclass $ node object_api/object_api_17.js
{ name: 'JavaScript', year: 1995, paradigm: 'OO and Functional' }
false
true
true
rodrigobranas:javascriptmasterclass $
```



CAUTION

Não é possível alterar o protótipo do  
objeto, que se torna imutável

A screenshot of a macOS desktop environment showing a terminal window. The title bar of the terminal window reads "object\_api\_18.js — javascriptmasterclass". The main pane of the terminal displays the contents of the file "object\_api\_18.js". The code defines a constant "javascript" as an object with properties: name ("JavaScript"), year (1995), and paradigm ("OO and Functional"). It then uses Object.freeze() to freeze the object. Subsequent code attempts to modify the object by changing its name to "ECMAScript", setting its author to "Brendan Eich", and deleting its year property. It also checks if the object is extensible, sealed, or frozen. Finally, it uses Object.setPrototypeOf() to set the prototype of the object to an empty object. The right pane of the terminal shows the output of running this script with "node". The output shows the initial state of the object, followed by several "true" responses to console.log() calls, and then the error message: "TypeError: #<Object> is not extensible". The stack trace for this error is provided, detailing the call stack from the original call to Object.setPrototypeOf() down to the final module load.

```
JS object_api_18.js x
object_api_18.js — javascriptmasterclass
1 const javascript = {
2   name: "JavaScript",
3   year: 1995,
4   paradigm: "OO and Functional"
5 };
6 Object.freeze(javascript);
7 javascript.name = "ECMAScript";
8 javascript.author = "Brendan Eich";
9 delete javascript.year;
10 console.log(javascript);
11 console.log(Object.isExtensible(javascript));
12 console.log(Object.isSealed(javascript));
13 console.log(Object.isFrozen(javascript));
14 Object.setPrototypeOf(javascript, {});
15

TERMINAL ... 1: bash + = 1
rodrigobranas:javascriptmasterclass $ node object_api/object_api_18.js
{ name: 'JavaScript', year: 1995, paradigm: 'OO and Functional' }
false
true
true
/Users/rodrigobranas/development/workspace/javascriptmasterclass/object_api/object_api_18.js:14
Object.setPrototypeOf(javascript, {});
^

TypeError: #<Object> is not extensible
    at Function.setPrototypeOf (<anonymous>)
    at Object.<anonymous> (/Users/rodrigobranas/development/workspace/javascriptmasterclass/object_api/object_api_18.js:14:8)
    at Module._compile (module.js:643:30)
    at Object.Module._extensions..js (module.js:654:10)
    at Module.load (module.js:556:32)
    at tryModuleLoad (module.js:499:12)
    at Function.Module._load (module.js:491:3)
    at Function.Module.runMain (module.js:684:10)
    at startup (bootstrap_node.js:187:16)
    at bootstrap_node.js:608:3
rodrigobranas:javascriptmasterclass $
```