

# Iterables e Iterators

São **convenções** implementadas por Arrays, Maps, Sets e Strings que os tornam iteráveis por meio de um protocolo de iteração

A screenshot of a macOS desktop environment showing a terminal window. The window title is "iterables\_iterators\_1.js — javascriptmasterclass". The main pane displays a block of JavaScript code:

```
JS iterables_iterators_1.js x
1 const languages = ["Fortran", "Lisp", "COBOL"];
2 for (let i = 0; i < languages.length; i++) {
3   console.log(languages[i]);
4 }
5
```

The code defines an array of three programming languages and uses a for loop to log each language to the console. The right pane shows the terminal output:

```
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_1.js
Fortran
Lisp
COBOL
rodrigobranas:javascriptmasterclass $
```

A screenshot of a terminal window titled "iterables\_iterators\_2.js — javascriptmasterclass". The window is divided into two main sections: a code editor on the left and a terminal on the right.

The code editor contains the following JavaScript code:

```
JS iterables_iterators_2.js ×
1 const languages = ["Fortran", "Lisp", "COBOL"];
2 for (let i in languages) {
3     console.log(languages[i]);
4 }
5
```

The terminal section shows the output of running the script:

```
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_2.js
Fortran
Lisp
COBOL
rodrigobranas:javascriptmasterclass $
```

A screenshot of a terminal window titled "iterables\_iterators\_3.js — javascriptmasterclass". The window is divided into two main sections: a code editor on the left and a terminal on the right.

The code editor contains the following JavaScript code:

```
JS iterables_iterators_3.js ×
1 const languages = ["Fortran", "Lisp", "COBOL"];
2 languages.forEach(function (language) {
3   console.log(language);
4 });
5
```

The terminal section shows the output of running the script:

```
rodrigobranas:javascriptmasterclass $ node i
terables_iterators/iterables_iterators_3.js
Fortran
Lisp
COBOL
rodrigobranas:javascriptmasterclass $ █
```

A screenshot of a terminal window titled "iterables\_iterators\_4.js — javascriptmasterclass". The window is divided into two main sections: a code editor on the left and a terminal on the right.

The code editor contains the following JavaScript code:

```
JS iterables_iterators_4.js ×
1 const languages = ["Fortran", "Lisp", "COBOL"];
2 languages.forEach((language) => {
3   console.log(language);
4 });
5
```

The terminal section shows the output of running the script with "node". The output is:

```
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_4.js
Fortran
Lisp
COBOL
rodrigobranas:javascriptmasterclass $
```

A screenshot of a macOS desktop environment showing a terminal window. The window title is "iterables\_iterators\_5.js — javascriptmasterclass". The main pane contains a code editor with a single file named "iterables\_iterators\_5.js". The code is as follows:

```
1 const languages = ["Fortran", "Lisp", "COBOL"];
2 for (let language of languages) {
3     console.log(language);
4 }
5
```

The terminal pane shows the output of running the script with the command "node iterables\_iterators/iterables\_iterators\_5.js". The output is:

```
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_5.js
Fortran
Lisp
COBOL
rodrigobranas:javascriptmasterclass $
```

A screenshot of a macOS desktop environment showing a terminal window. The window title is "iterables\_iterators\_6.js — javascriptmasterclass". The main pane contains a JavaScript code editor with the file "iterables\_iterators\_6.js". The code defines three arrays: "classicLanguages", "modernLanguages", and "languages" which is the concatenation of the two. It then logs "languages" to the console. The output pane shows the command "node iterables\_iterators/iterables\_iterators\_6.js" being run, followed by the returned array [ 'Fortran', 'Lisp', 'COBOL', 'Python', 'Ruby', 'JavaScript' ].

```
JS iterables_iterators_6.js × iterables_iterators_6.js — javascriptmasterclass TERMINAL ... 1: bash + ⌂ 1

1 const classicLanguages = ["Fortran", "Lisp", "COBOL"];
2 const modernLanguages = ["Python", "Ruby", "JavaScript"]
3 const languages = classicLanguages.concat(modernLanguages);
4 console.log(languages);
5

rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_6.js
[ 'Fortran', 'Lisp', 'COBOL', 'Python', 'Ruby', 'JavaScript' ]
rodrigobranas:javascriptmasterclass $
```

iterables\_iterators\_7.js — javascriptmasterclass

JS iterables\_iterators\_7.js x

TERMINAL ... 1: bash + =

```
1 const classicLanguages = ["Fortran", "Lisp", "COBOL"];
2 const modernLanguages = ["Python", "Ruby", "JavaScript"]
3 const languages = [...classicLanguages, ...modernLanguages];
4 console.log(languages);
5
```

rodrigobranas:javascriptmasterclass \$ node iterables\_iterators/iterables\_iterators\_7.js  
[ 'Fortran', 'Lisp', 'COBOL', 'Python', 'Ruby', 'JavaScript' ]  
rodrigobranas:javascriptmasterclass \$

Além do Array é possível utilizar o protocolo de iteração dos objetos Map, Set e String

iterables\_iterators\_8.js — javascriptmasterclass

JS iterables\_iterators\_8.js ×

TERMINAL ... 1: bash + ⌂

```
1 const languages = new Map([["Fortran", 1957], ["Lisp", 1958])
2 for (let language of languages) {
3   console.log(language);
4 }
5
```

rodrigobranas:javascriptmasterclass \$ node iterables\_iterators/iterables\_iterators\_8.js

[ 'Fortran', 1957 ]  
[ 'Lisp', 1958 ]  
[ 'COBOL', 1959 ]

rodrigobranas:javascriptmasterclass \$

A screenshot of a terminal window titled "iterables\_iterators\_9.js — javascriptmasterclass". The window is divided into two main sections: a code editor on the left and a terminal on the right.

The code editor contains the following JavaScript code:

```
JS iterables_iterators_9.js ×
1 const languages = new Map([["Fortran", 1957], ["Lisp", 1958]]);
2 for (let [language, year] of languages) {
3     console.log(language, year);
4 }
5
```

The terminal section shows the output of running the script with "node".

```
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_9.js
Fortran 1957
Lisp 1958
COBOL 1959
rodrigobranas:javascriptmasterclass $
```

iterables\_iterators\_10.js — javascriptmasterclass

JS iterables\_iterators\_10.js ×

TERMINAL ... 1: bash + ⌂

```
1 const languages = new Map([["Fortran", 1957], ["Lisp", 1958]]);  
2 console.log([...languages]);  
3
```

rodrigobranas:javascriptmasterclass \$ node iterables\_iterators/iterables\_iterators\_10.js  
[ [ 'Fortran', 1957 ], [ 'Lisp', 1958 ], [ 'COBOL', 1959 ] ]  
rodrigobranas:javascriptmasterclass \$

A screenshot of a macOS desktop environment showing a terminal window. The window title is "iterables\_iterators\_11.js — javascriptmasterclass". The main pane contains a code editor with a single file named "iterables\_iterators\_11.js". The code is as follows:

```
1 let languages = new Set(["Fortran", "Lisp", "COBOL"]);
2 for (let language of languages) {
3     console.log(language);
4 }
5
```

The terminal pane shows the output of running the script with the command "node iterables\_iterators/iterables\_iterators\_11.js". The output is:

```
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_11.js
Fortran
Lisp
COBOL
rodrigobranas:javascriptmasterclass $
```

A screenshot of a terminal window titled "iterables\_iterators\_12.js — javascriptmasterclass". The window is divided into two main sections: a code editor on the left and a terminal on the right.

The code editor contains the following JavaScript code:

```
JS iterables_iterators_12.js ×
1 let languages = new Set(["Fortran", "Lisp", "COBOL"]);
2 console.log([...languages]);
3
```

The terminal section shows the output of running the script:

```
TERMINAL ... 1: bash
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_12.js
[ 'Fortran', 'Lisp', 'COBOL' ]
rodrigobranas:javascriptmasterclass $
```

A screenshot of a terminal window titled "iterables\_iterators\_13.js — javascriptmasterclass". The window is divided into two main sections: a code editor on the left and a terminal on the right.

The code editor contains the following JavaScript code:

```
JS iterables_iterators_13.js ×
1 let language = "COBOL";
2 for (let char of language) {
3     console.log(char);
4 }
5
```

The terminal section shows the output of running the script with the command "node iterables\_iterators/iterables\_iterators\_13.js". The output is:

```
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_13.js
C
O
B
O
L
rodrigobranas:javascriptmasterclass $
```

A screenshot of a macOS desktop environment showing a terminal window. The window title is "iterables\_iterators\_14.js — javascriptmasterclass". The main pane contains the following code:

```
JS iterables_iterators_14.js ×
1 let language = "COBOL";
2 console.log([...language]);
3
```

The terminal pane shows the output of running the script with Node.js:

```
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_14.js
[ 'C', 'O', 'B', 'O', 'L' ]
rodrigobranas:javascriptmasterclass $
```

Todo **Iterable** tem um propriedade de chave `Symbol.iterator` que define o protocolo de iteração para o objeto

A screenshot of a macOS desktop environment showing a terminal window. The title bar of the window reads "iterables\_iterators\_15.js — javascriptmasterclass". The main area contains a code editor with a single file named "iterables\_iterators\_15.js". The code defines a constant "languages" array containing three elements: "Fortran", "Lisp", and "COBOL". It then creates an iterator for this array using the Symbol.iterator() method and logs four consecutive calls to the iterator's next() method to the console. The output in the terminal shows the first three results as objects with "value" and "done" properties, followed by the fourth result which has "value" as undefined and "done" as true, indicating the end of the iteration.

```
JS iterables_iterators_15.js × iterables_iterators_15.js — javascriptmasterclass TERMINAL ... 1: bash + ⌂ 1  
1 const languages = ["Fortran", "Lisp", "COBOL"];  
2 const iterator = languages[Symbol.iterator]();  
3 console.log(iterator.next());  
4 console.log(iterator.next());  
5 console.log(iterator.next());  
6 console.log(iterator.next());  
7  
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_15.js  
{ value: 'Fortran', done: false }  
{ value: 'Lisp', done: false }  
{ value: 'COBOL', done: false }  
{ value: undefined, done: true }  
rodrigobranas:javascriptmasterclass $
```

iterables\_iterators\_16.js — javascriptmasterclass

JS iterables\_iterators\_16.js ×

TERMINAL ... 1: bash + ⌂ 1

```
1 const languages = new Map([["Fortran", 1957], ["Lisp", 1958]]
2 const iterator = languages[Symbol.iterator]()
3 console.log(iterator.next());
4 console.log(iterator.next());
5 console.log(iterator.next());
6 console.log(iterator.next());
7
```

rodrigobranas:javascriptmasterclass \$ node iterables\_iterators/iterables\_iterators\_16.js

```
{ value: [ 'Fortran', 1957 ], done: false }
{ value: [ 'Lisp', 1958 ], done: false }
{ value: [ 'COBOL', 1959 ], done: false }
{ value: undefined, done: true }
```

rodrigobranas:javascriptmasterclass \$

A screenshot of a terminal window titled "iterables\_iterators\_17.js — javascriptmasterclass". The window is divided into two main sections: a code editor on the left and a terminal on the right.

The code editor contains the following JavaScript code:

```
1 const languages = new Map([["Fortran", 1957], ["Lisp", 1958]]);  
2 const iterator = languages.entries();  
3 console.log(iterator.next());  
4 console.log(iterator.next());  
5 console.log(iterator.next());  
6 console.log(iterator.next());  
7
```

The terminal section shows the output of running the script with "node".

```
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_17.js  
{ value: [ 'Fortran', 1957 ], done: false }  
{ value: [ 'Lisp', 1958 ], done: false }  
{ value: [ 'COBOL', 1959 ], done: false }  
{ value: undefined, done: true }  
rodrigobranas:javascriptmasterclass $
```

A screenshot of a macOS desktop environment showing a terminal window and a code editor. The terminal window is titled 'iterables\_iterators\_18.js — javascriptmasterclass' and contains the command 'node iterables\_iterators/iterables\_iterators\_18.js'. The output shows four objects returned by the iterator's next() method, each with a 'value' property and a 'done' property set to false. After the fourth iteration, the 'done' property is set to true.

iterables\_iterators\_18.js — javascriptmasterclass

JS iterables\_iterators\_18.js × TERMINAL ... 1: bash + ⌂

```
1 const languages = new Map([["Fortran", 1957], ["Lisp", 1958]]);  
2 const iterator = languages.keys();  
3 console.log(iterator.next());  
4 console.log(iterator.next());  
5 console.log(iterator.next());  
6 console.log(iterator.next());  
7
```

```
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_18.js  
{ value: 'Fortran', done: false }  
{ value: 'Lisp', done: false }  
{ value: 'COBOL', done: false }  
{ value: undefined, done: true }  
rodrigobranas:javascriptmasterclass $
```

A screenshot of a terminal window titled "iterables\_iterators\_19.js — javascriptmasterclass". The window is divided into several sections: a top bar with red, yellow, and green buttons; a tab bar with "JS iterables\_iterators\_19.js" and a close button; a toolbar with icons for search, refresh, and more; a main code editor area containing the following JavaScript code; and a terminal output area.

```
1 const languages = new Map([["Fortran", 1957], ["Lisp", 1958]]);  
2 const iterator = languages.values();  
3 console.log(iterator.next());  
4 console.log(iterator.next());  
5 console.log(iterator.next());  
6 console.log(iterator.next());  
7
```

The terminal output shows the execution of the script:

```
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_19.js  
{ value: 1957, done: false }  
{ value: 1958, done: false }  
{ value: 1959, done: false }  
{ value: undefined, done: true }  
rodrigobranas:javascriptmasterclass $
```

A screenshot of a macOS desktop environment showing a terminal window. The title bar of the window reads "iterables\_iterators\_20.js — javascriptmasterclass". The main area contains a code editor with a single file named "iterables\_iterators\_20.js". The code defines a Set of three languages and creates an iterator for it, then logs each iteration's result to the console. The terminal pane to the right shows the output of running the script with "node".

```
JS iterables_iterators_20.js × iterables_iterators_20.js — javascriptmasterclass
1 const languages = new Set(["Fortran", "Lisp", "COBOL"]);
2 const iterator = languages[Symbol.iterator]();
3 console.log(iterator.next());
4 console.log(iterator.next());
5 console.log(iterator.next());
6 console.log(iterator.next());
7

TERMINAL ... 1: bash + ⌂ 1
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_20.js
{ value: 'Fortran', done: false }
{ value: 'Lisp', done: false }
{ value: 'COBOL', done: false }
{ value: undefined, done: true }
rodrigobranas:javascriptmasterclass $
```

A screenshot of a macOS desktop environment showing a terminal window. The title bar of the window reads "iterables\_iterators\_21.js — javascriptmasterclass". The main area contains a code editor with a single file named "iterables\_iterators\_21.js". The code defines a Set of three languages and creates an iterator for it, then logs each entry and the final undefined value. To the right of the code editor is a terminal pane titled "1: bash" which shows the output of running the script with node. The output consists of four objects, each representing a step in the iteration of the Set, followed by a final object indicating the iteration is done.

```
JS iterables_iterators_21.js × iterables_iterators_21.js — javascriptmasterclass
1 const languages = new Set(["Fortran", "Lisp", "COBOL"]);
2 const iterator = languages.entries();
3 console.log(iterator.next());
4 console.log(iterator.next());
5 console.log(iterator.next());
6 console.log(iterator.next());
7

rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_21.js
{ value: [ 'Fortran', 'Fortran' ], done: false }
{ value: [ 'Lisp', 'Lisp' ], done: false }
{ value: [ 'COBOL', 'COBOL' ], done: false }
{ value: undefined, done: true }
rodrigobranas:javascriptmasterclass $
```

A screenshot of a macOS desktop environment showing a terminal window. The title bar of the window reads "iterables\_iterators\_22.js — javascriptmasterclass". The main area contains a code editor with a single file named "iterables\_iterators\_22.js". The code defines a Set of three languages and creates an iterator for it, then logs each iteration's value and done status. The terminal pane to the right shows the output of running the script with "node".

```
JS iterables_iterators_22.js × iterables_iterators_22.js — javascriptmasterclass
1 const languages = new Set(["Fortran", "Lisp", "COBOL"]);
2 const iterator = languages.keys();
3 console.log(iterator.next());
4 console.log(iterator.next());
5 console.log(iterator.next());
6 console.log(iterator.next());
7

TERMINAL ... 1: bash + ⌂
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_22.js
{ value: 'Fortran', done: false }
{ value: 'Lisp', done: false }
{ value: 'COBOL', done: false }
{ value: undefined, done: true }
rodrigobranas:javascriptmasterclass $
```

iterables\_iterators\_23.js — javascriptmasterclass

JS iterables\_iterators\_23.js x

TERMINAL ... 1: bash + =

```
1 const languages = new Set(["Fortran", "Lisp", "COBOL"]);
2 const iterator = languages.values();
3 console.log(iterator.next());
4 console.log(iterator.next());
5 console.log(iterator.next());
6 console.log(iterator.next());
7
```

rodrigobranas:javascriptmasterclass \$ node iterables\_iterators/iterables\_iterators\_23.js

{ value: 'Fortran', done: false }  
{ value: 'Lisp', done: false }  
{ value: 'COBOL', done: false }  
{ value: undefined, done: true }

rodrigobranas:javascriptmasterclass \$

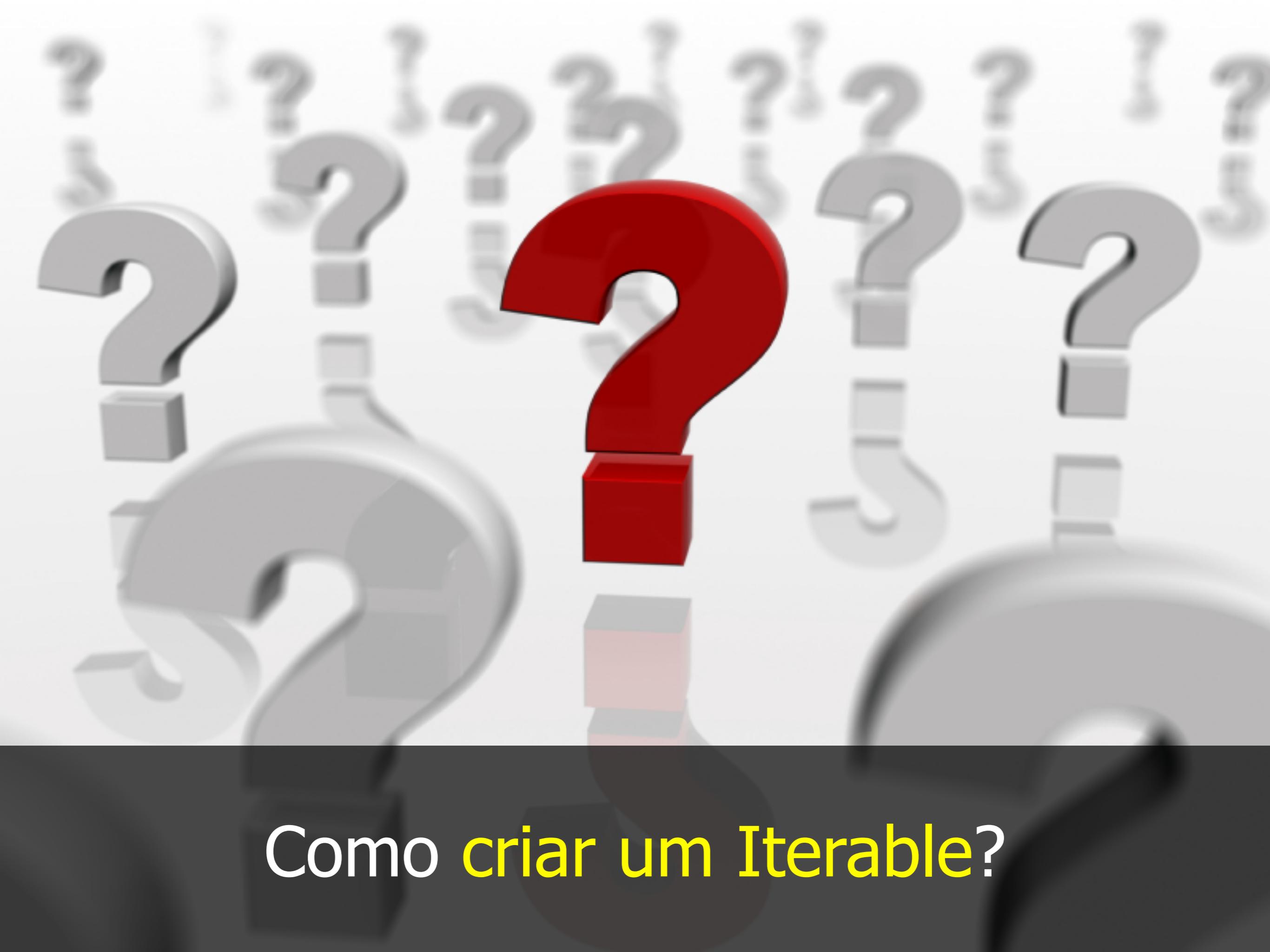
A screenshot of a macOS desktop environment showing a terminal window and a code editor. The terminal window is titled 'iterables\_iterators\_24.js — javascriptmasterclass' and contains the command 'node iterables\_iterators/iterables\_iterators\_24.js'. The output of the script is displayed below, showing the iteration of the string 'COBOL' using a generator function.

The code editor window shows the following JavaScript code:

```
JS iterables_iterators_24.js x
1 const language = "COBOL";
2 const iterator = language[Symbol.iterator]();
3 console.log(iterator.next());
4 console.log(iterator.next());
5 console.log(iterator.next());
6 console.log(iterator.next());
7 console.log(iterator.next());
8 console.log(iterator.next());
9
```

The terminal output is:

```
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_24.js
{ value: 'C', done: false }
{ value: 'O', done: false }
{ value: 'B', done: false }
{ value: 'O', done: false }
{ value: 'L', done: false }
{ value: undefined, done: true }
rodrigobranas:javascriptmasterclass $
```



Como criar um Iterable?

A screenshot of a terminal window showing the execution of a JavaScript iterator function. The terminal window is titled "iterables\_iterators\_25.js — javascriptmasterclass". The code in the editor is as follows:

```
JS iterables_iterators_25.js x
1  function createIterator(...array) {
2      let i = 0;
3      return {
4          next() {
5              if (i < array.length) {
6                  return {
7                      value: array[i++],
8                      done: false
9                  }
10             } else {
11                 return {
12                     value: undefined,
13                     done: true
14                 }
15             }
16         }
17     }
18 }
19 const iterator = createIterator("Fortran", "Lisp", "COBOL");
20 console.log(iterator.next());
21 console.log(iterator.next());
22 console.log(iterator.next());
23 console.log(iterator.next());
24
```

The terminal output shows the results of calling the `next` method four times on the iterator object:

```
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_25.js
{ value: 'Fortran', done: false }
{ value: 'Lisp', done: false }
{ value: 'COBOL', done: false }
{ value: undefined, done: true }
rodrigobranas:javascriptmasterclass $
```

A screenshot of a terminal window titled "iterables\_iterators\_26.js — javascriptmasterclass". The terminal shows the following output:

```
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_26.js
/Users/rodrigobranas/development/workspace/javascriptmasterclass/iterables_iterators/iterables_iterators_26.js:20
for (let language of iterator) {
^

TypeError: iterator is not iterable
    at Object.<anonymous> (/Users/rodrigobranas/development/workspace/javascriptmasterclass/iterables_iterators/iterables_iterators_26.js:20:22)
        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 $
```

The code in the editor is as follows:

```
1  function createIterator(...array) {
2      let i = 0;
3      return {
4          next() {
5              if (i < array.length) {
6                  return {
7                      value: array[i++],
8                      done: false
9                  }
10             } else {
11                 return {
12                     value: undefined,
13                     done: true
14                 }
15             }
16         }
17     }
18 }
19 const iterator = createIterator("Fortran", "Lisp", "COBOL");
20 for (let language of iterator) {
21     console.log(language);
22 }
```

A screenshot of a terminal window titled "iterables\_iterators\_27.js — javascriptmasterclass". The terminal shows the following output:

```
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_27.js
/Users/rodrigobranas/development/workspace/javascriptmasterclass/iterables_iterators/iterables_iterators_27.js:20
  console.log([...iterator]);
               ^
TypeError: iterator is not iterable
    at Object.<anonymous> (/Users/rodrigobranas/development/workspace/javascriptmasterclass/iterables_iterators/iterables_iterators_27.js:20:17)
        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 $
```

The terminal window is part of a larger interface, likely a code editor, showing the file "iterables\_iterators\_27.js" on the left. The code defines a function "createIterator" that returns an iterator object. The iterator's "next" method checks if "i" is less than the array's length. If true, it returns an object with "value" set to the current element and "done" set to false. If false, it returns an object with "value" set to undefined and "done" set to true. The "console.log" statement at line 20 attempts to spread the iterator object, which results in the error "TypeError: iterator is not iterable".

A screenshot of a macOS desktop environment showing a terminal window and a code editor. The terminal window is titled 'iterables\_iterators\_28.js — javascriptmasterclass' and contains the command 'node iterables\_iterators/iterables\_iterators\_28.js'. The output shows three lines of text: 'Fortran', 'Lisp', and 'COBOL'. The code editor window is titled 'JS iterables\_iterators\_28.js' and displays the source code for the file. The code defines a function 'createIterable' that returns an iterator object. The iterator's 'next' method checks if the index 'i' is less than the array's length. If true, it returns an object with 'value' set to the current element and 'done' set to false. If false, it returns an object with 'value' set to undefined and 'done' set to true. The main part of the script creates an iterable object with the languages 'Fortran', 'Lisp', and 'COBOL' and then loops through it, logging each language to the console.

```
function createIterable(...array) {
  return {
    [Symbol.iterator]() {
      let i = 0;
      return {
        next() {
          if (i < array.length) {
            return {
              value: array[i++],
              done: false
            }
          } else {
            return {
              value: undefined,
              done: true
            }
          }
        }
      }
    }
  }
}

const iterable = createIterable("Fortran", "Lisp", "COBOL");
for (let language of iterable) {
  console.log(language);
}
```

A screenshot of a terminal window titled "iterables\_iterators\_29.js — javascriptmasterclass". The terminal shows the command "node iterables\_iterators/iterables\_iterators\_29.js" being run, followed by the output "[ 'Fortran', 'Lisp', 'COBOL' ]".

```
JS iterables_iterators_29.js × iterables_iterators_29.js — javascriptmasterclass TERMINAL ... 1: bash + ⌂ 1  
rodrigobranas:javascriptmasterclass $ node iterables_iterators/iterables_iterators_29.js  
[ 'Fortran', 'Lisp', 'COBOL' ]  
rodrigobranas:javascriptmasterclass $  
  
1  function createIterable(...array) {  
2      return {  
3          [Symbol.iterator]() {  
4              let i = 0;  
5              return {  
6                  next() {  
7                      if (i < array.length) {  
8                          return {  
9                              value: array[i++],  
10                             done: false  
11                         }  
12                     } else {  
13                         return {  
14                             value: undefined,  
15                             done: true  
16                         }  
17                     }  
18                 }  
19             }  
20         }  
21     }  
22 }  
23 const iterable = createIterable("Fortran", "Lisp", "COBOL");  
24 console.log([...iterable]);  
25
```