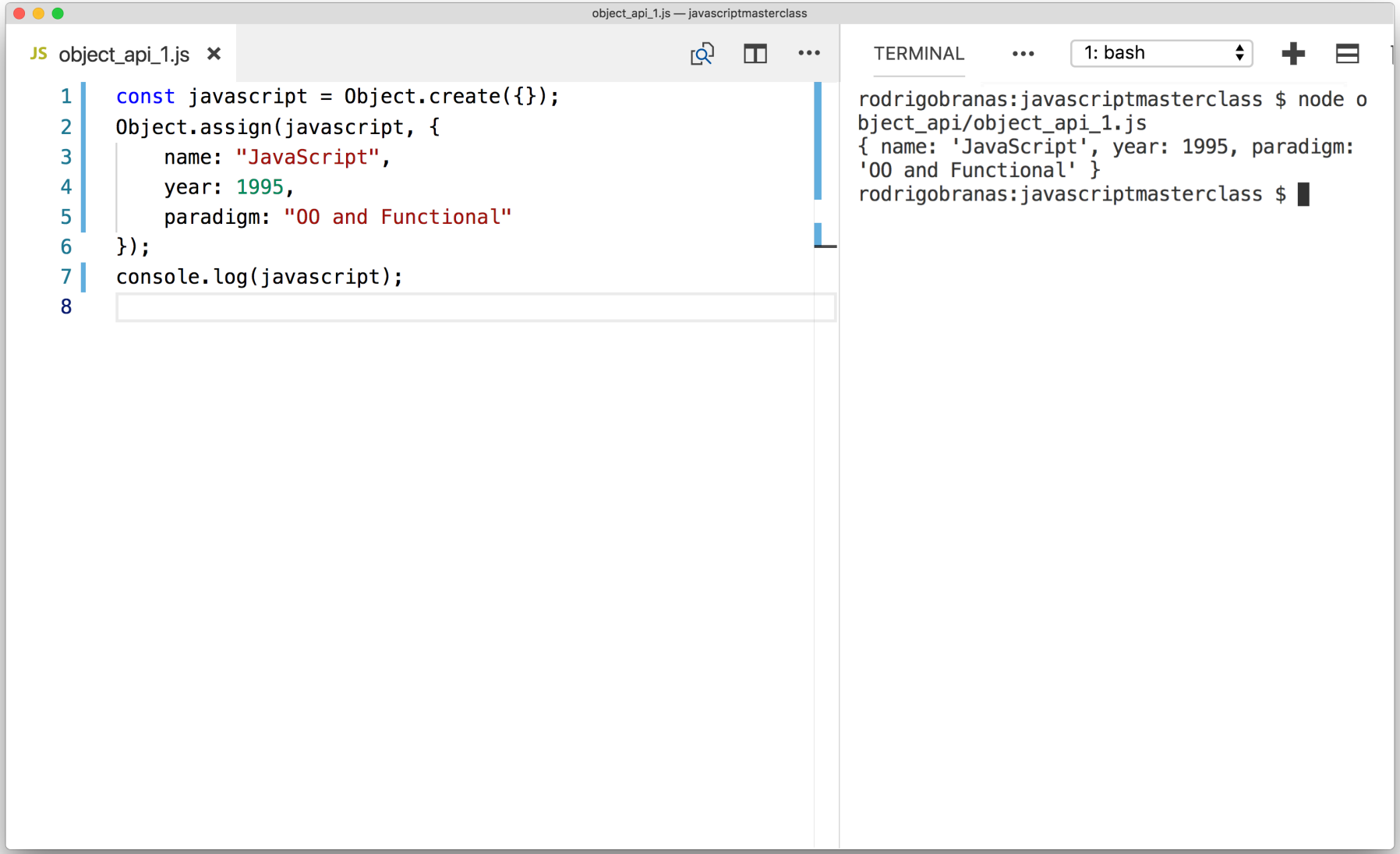
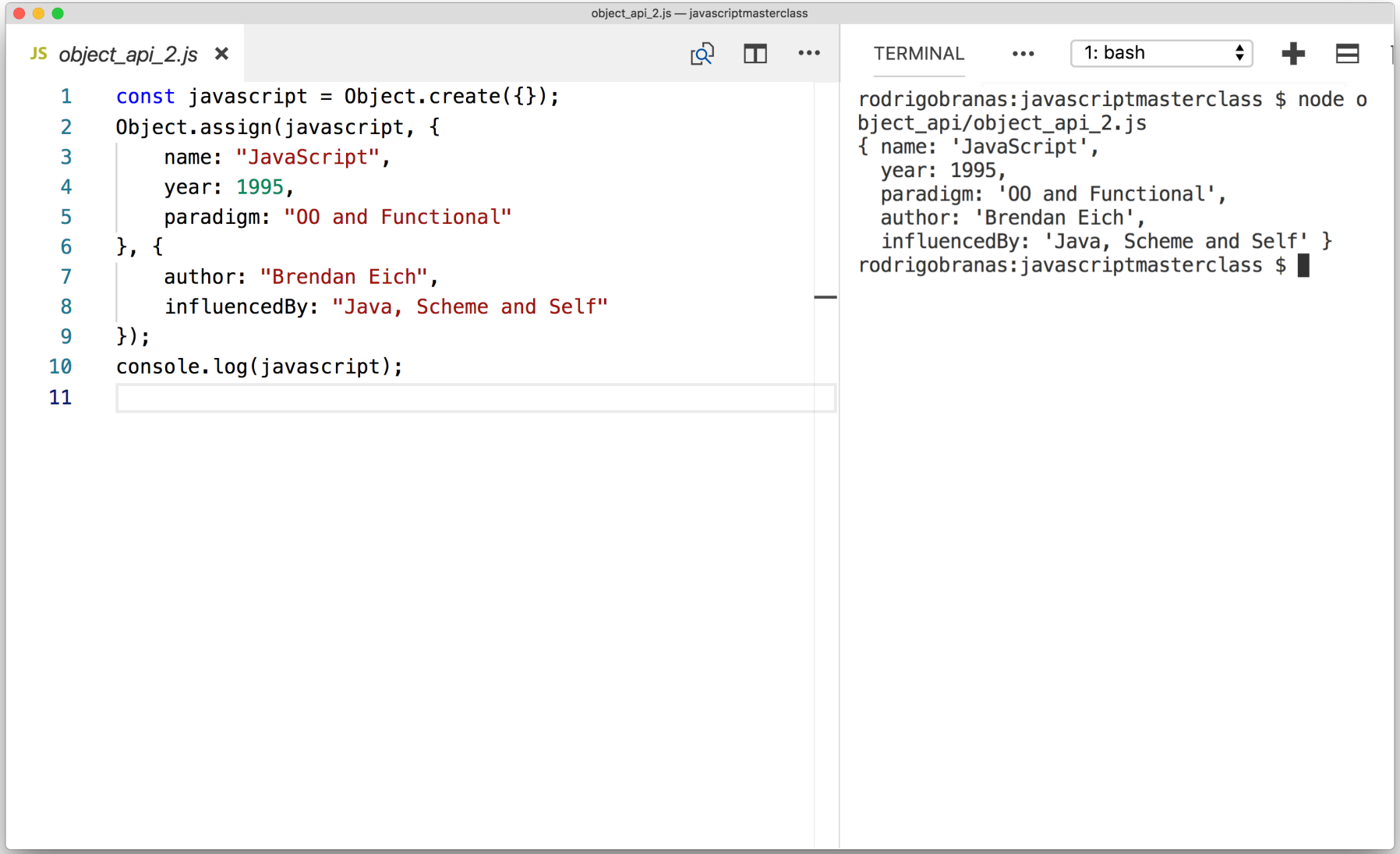
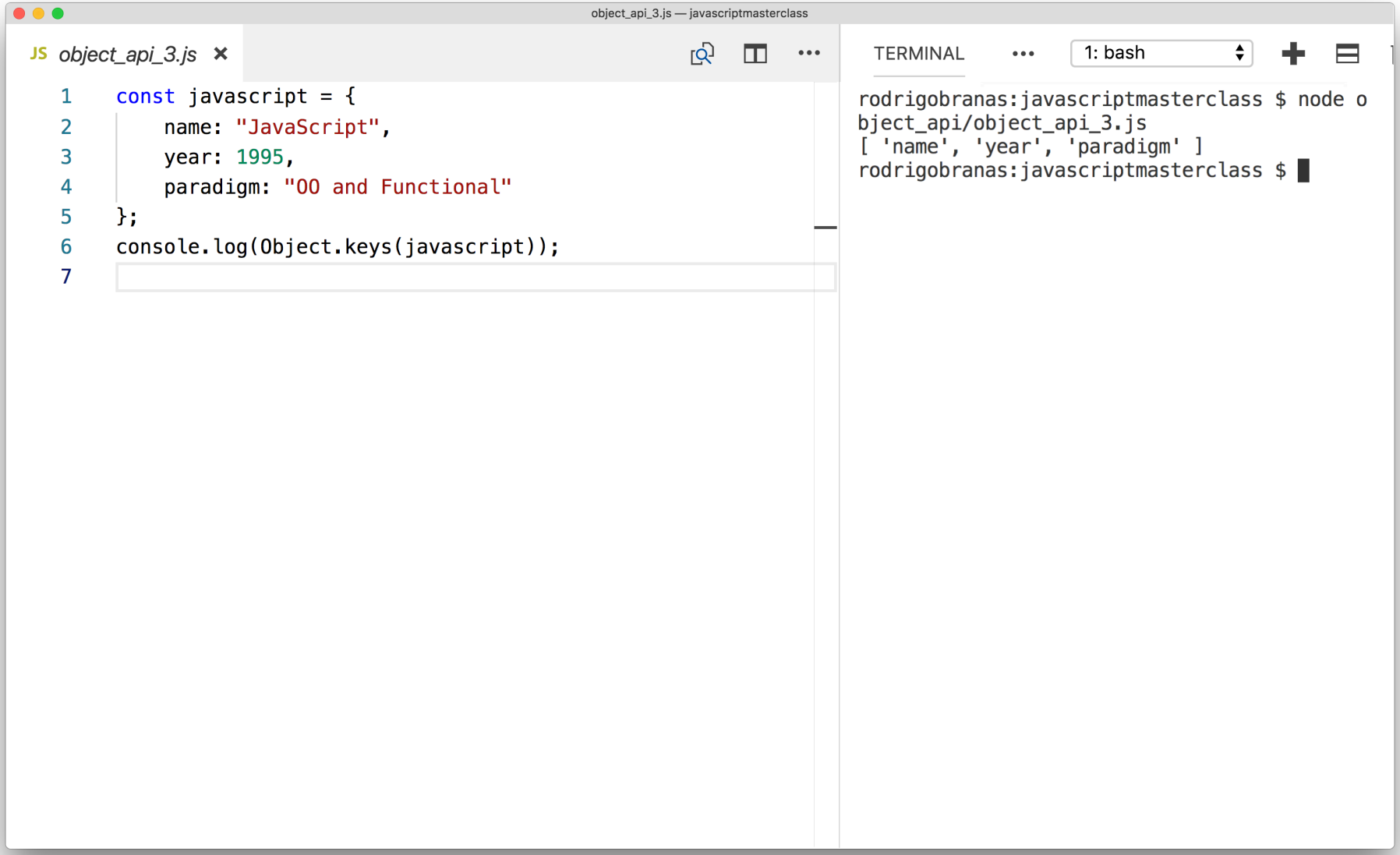

Object API

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

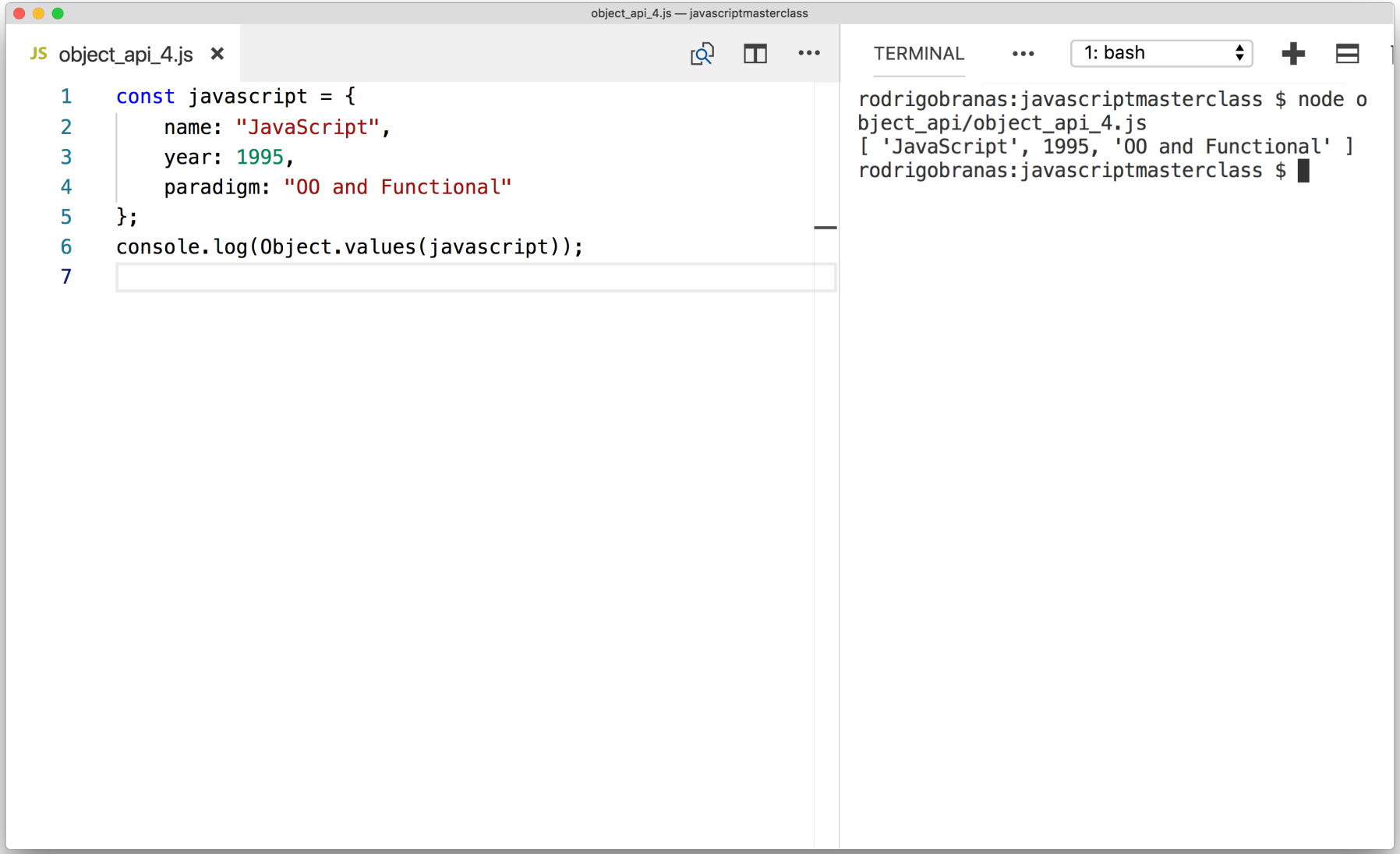

object_api_1.js — javascriptmasterclass
object_api_1.js
1 const javascript = Object.create({});
2 Object.assign(javascript, {
3 name: "JavaScript",
4 year: 1995,
5 paradigm: "OO and Functional"
6 });
7 console.log(javascript);
8
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node o
bject_api/object_api_1.js
{ name: 'JavaScript', year: 1995, paradigm:
'OO and Functional' }
rodrigobranas:javascriptmasterclass $


object_api_2.js — javascriptmasterclass
JS object_api_2.js
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 o
bject_api/object_api_2.js
{ name: 'JavaScript',
year: 1995,
paradigm: 'OO and Functional',
author: 'Brendan Eich',
influencedBy: 'Java, Scheme and Self' }
rodrigobranas:javascriptmasterclass $

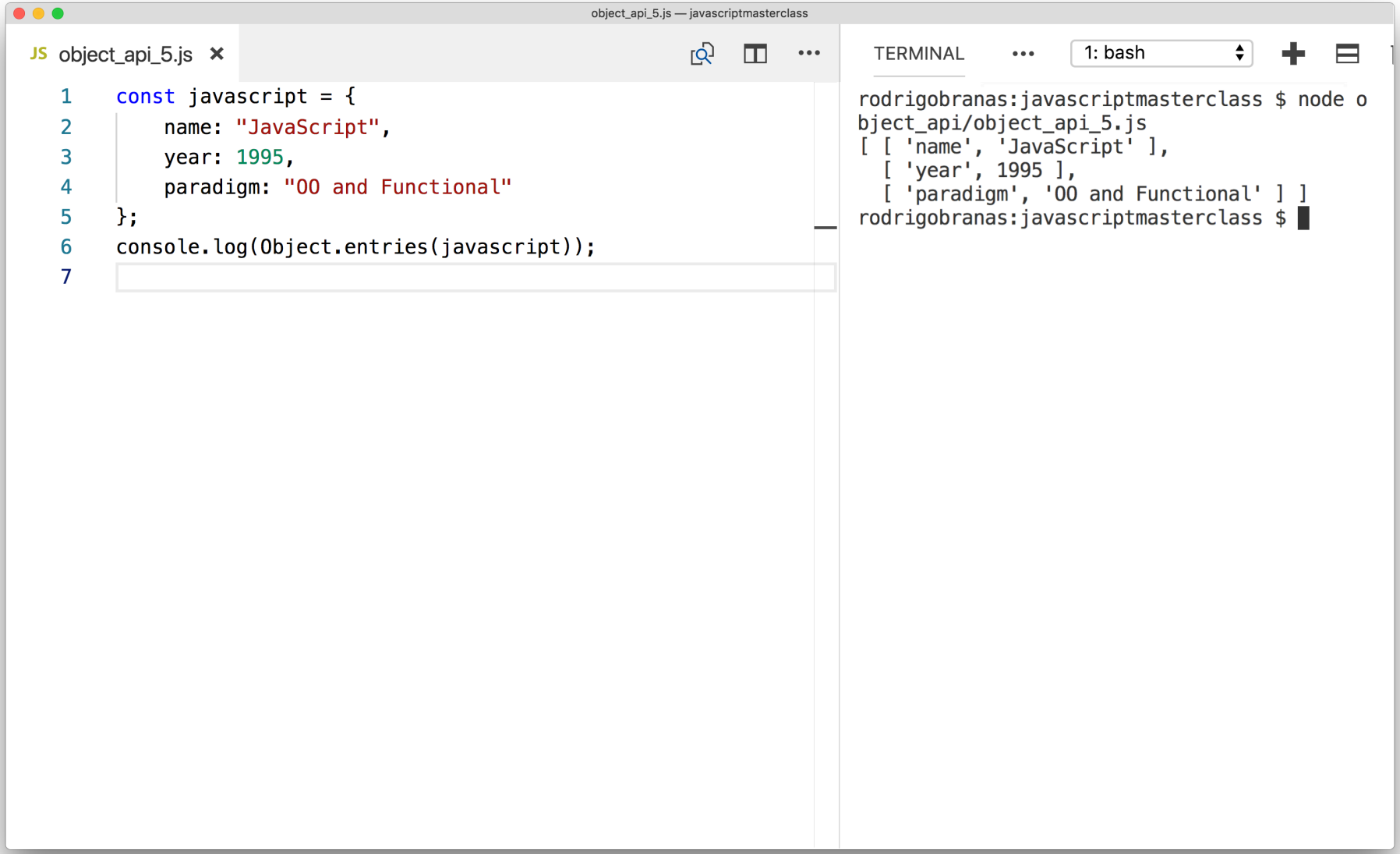
O método Object.keys retorna as chaves das propriedades do objeto


object_api_3.js — javascriptmasterclass
object_api_3.js
1 const javascript = {
2 name: "JavaScript",
3 year: 1995,
4 paradigm: "OO and Functional"
5 };
6 console.log(Object.keys(javascript));
7
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node o
bject_api/object_api_3.js
['name', 'year', 'paradigm']
rodrigobranas:javascriptmasterclass $

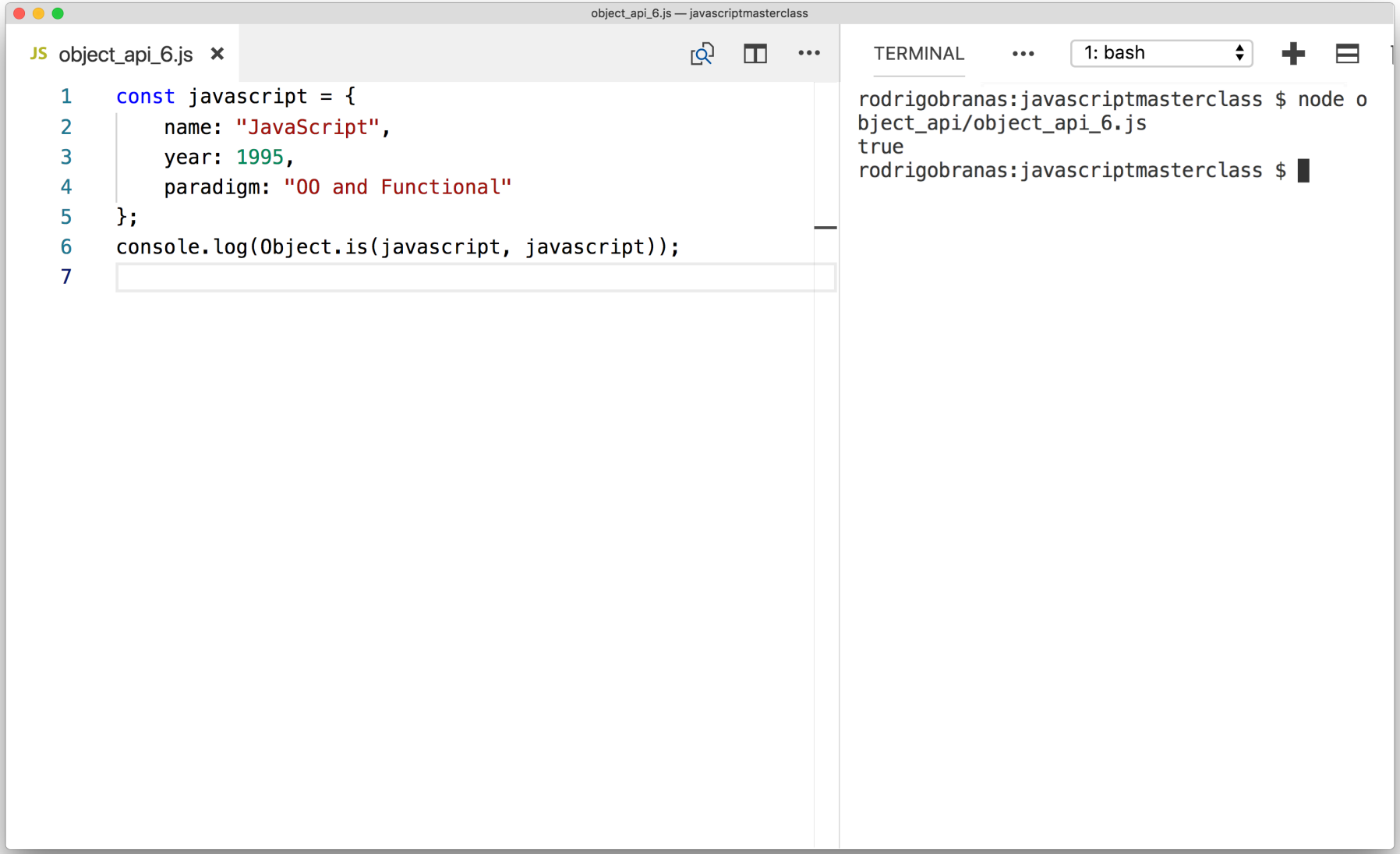
O método Object.values retorna os valores das propriedades do objeto

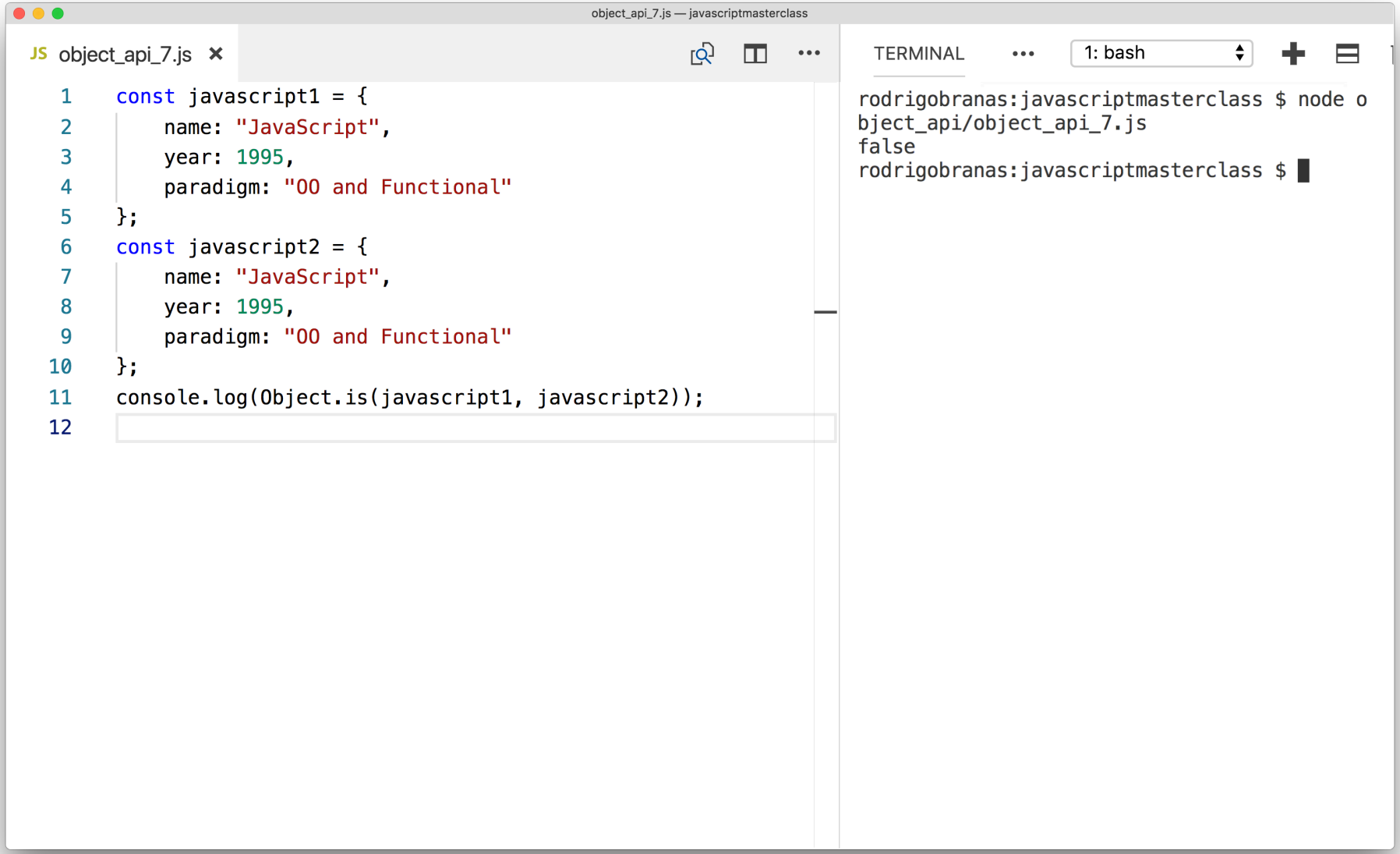

object_api_4.js — javascriptmasterclass
object_api_4.js
1 const javascript = {
2 name: "JavaScript",
3 year: 1995,
4 paradigm: "OO and Functional"
5 };
6 console.log(Object.values(javascript));
7
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node o
bject_api/object_api_4.js
['JavaScript', 1995, 'OO and Functional']
rodrigobranas:javascriptmasterclass $

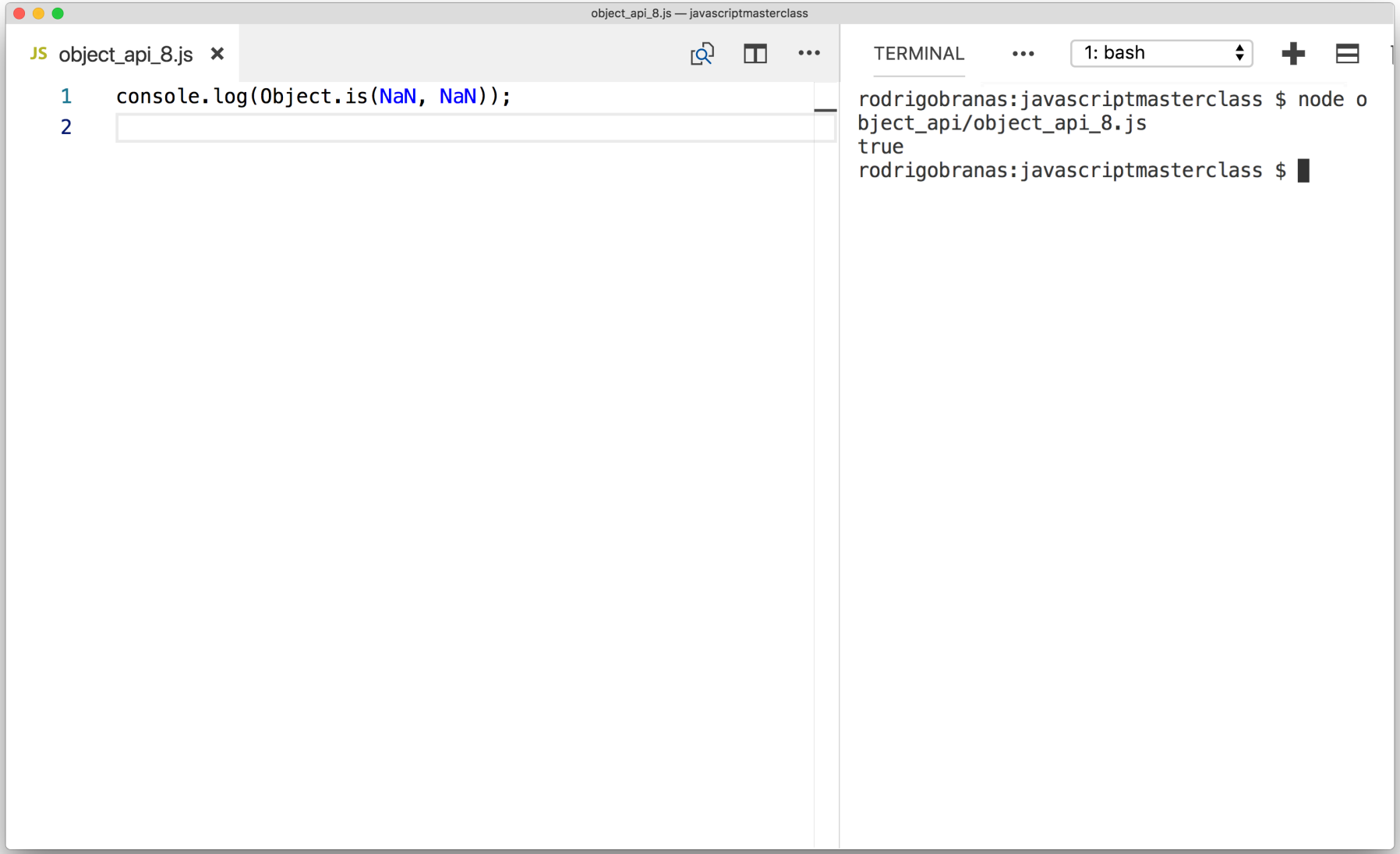
O método Object.entries retorna as propriedades do objeto em pares de chave e valor


object_api_5.js — javascriptmasterclass
object_api_5.js
1 const javascript = {
2 name: "JavaScript",
3 year: 1995,
4 paradigm: "OO and Functional"
5 };
6 console.log(Object.entries(javascript));
7
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node o
bject_api/object_api_5.js
[['name', 'JavaScript'],
['year', 1995],
['paradigm', 'OO and Functional']]
rodrigobranas:javascriptmasterclass $

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

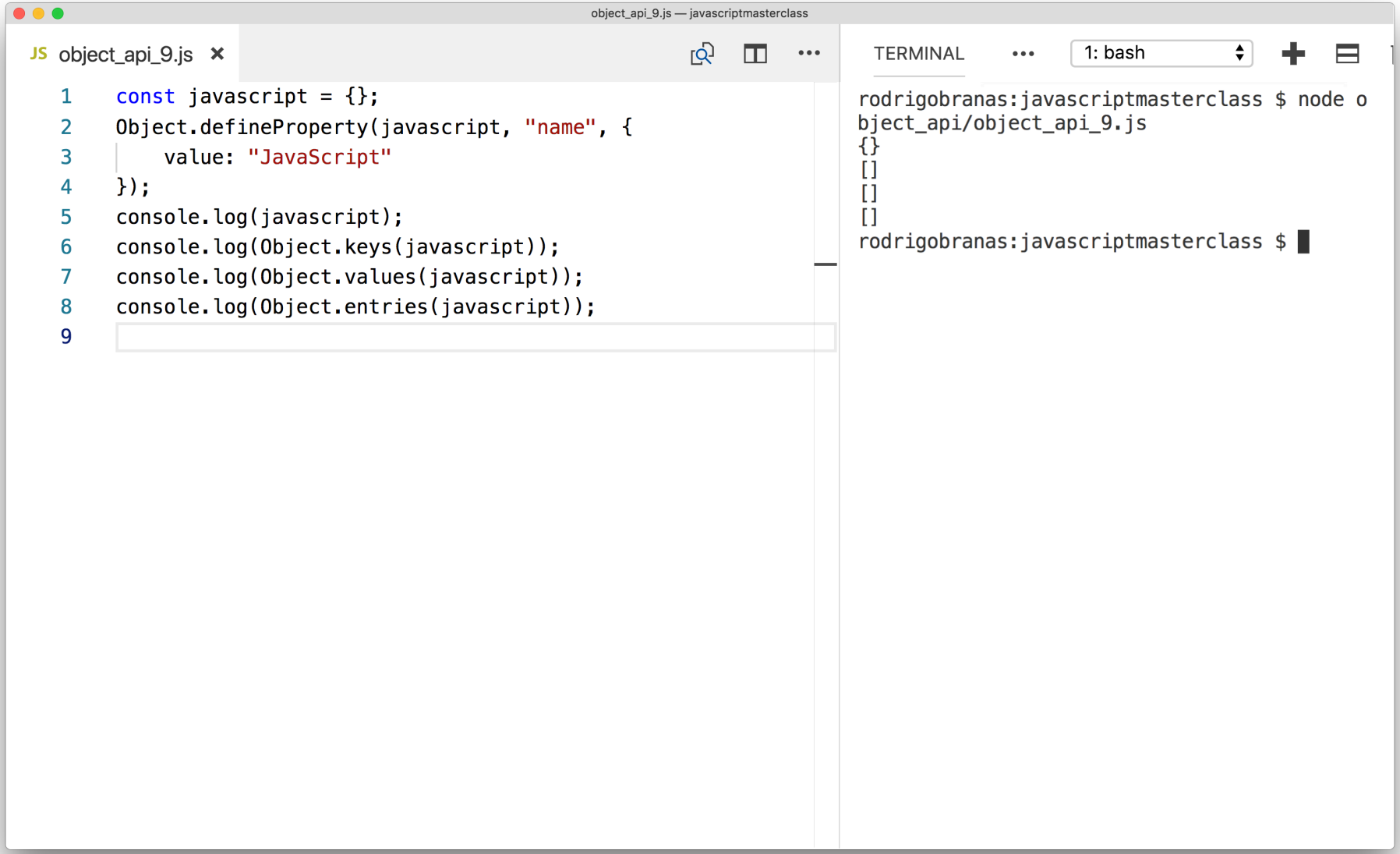

object_api_6.js — javascriptmasterclass
object_api_6.js
1 const javascript = {
2 name: "JavaScript",
3 year: 1995,
4 paradigm: "OO and Functional"
5 };
6 console.log(Object.is(javascript, javascript));
7
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node o
bject_api/object_api_6.js
true
rodrigobranas:javascriptmasterclass $

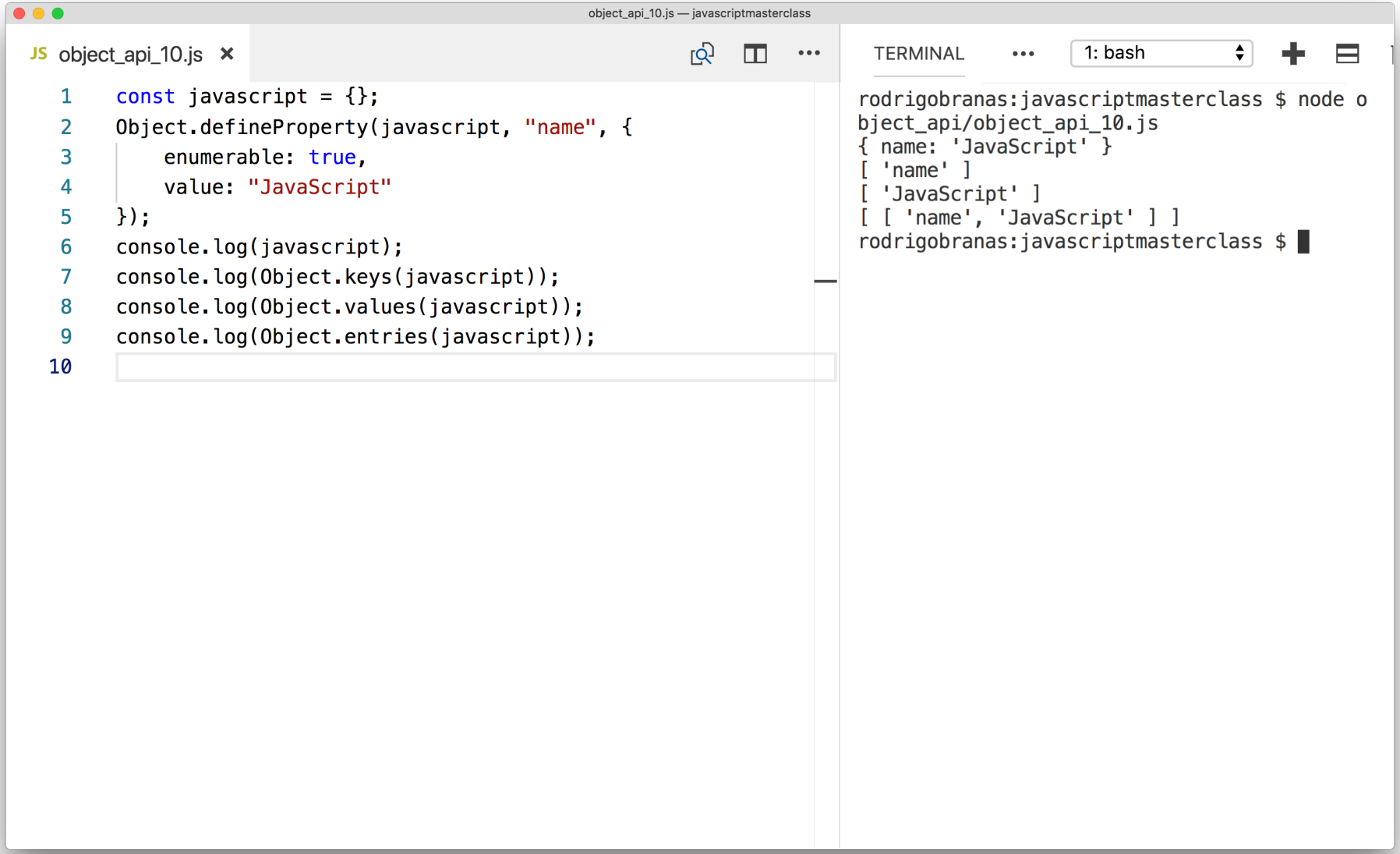

object_api_7.js — javascriptmasterclass
object_api_7.js
1 const javascript1 = {
2 name: "JavaScript",
3 year: 1995,
4 paradigm: "OO and Functional"
5 };
6 const javascript2 = {
7 name: "JavaScript",
8 year: 1995,
9 paradigm: "OO and Functional"
10 };
11 console.log(Object.is(javascript1, javascript2));
12
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node o
bject_api/object_api_7.js
false
rodrigobranas:javascriptmasterclass $

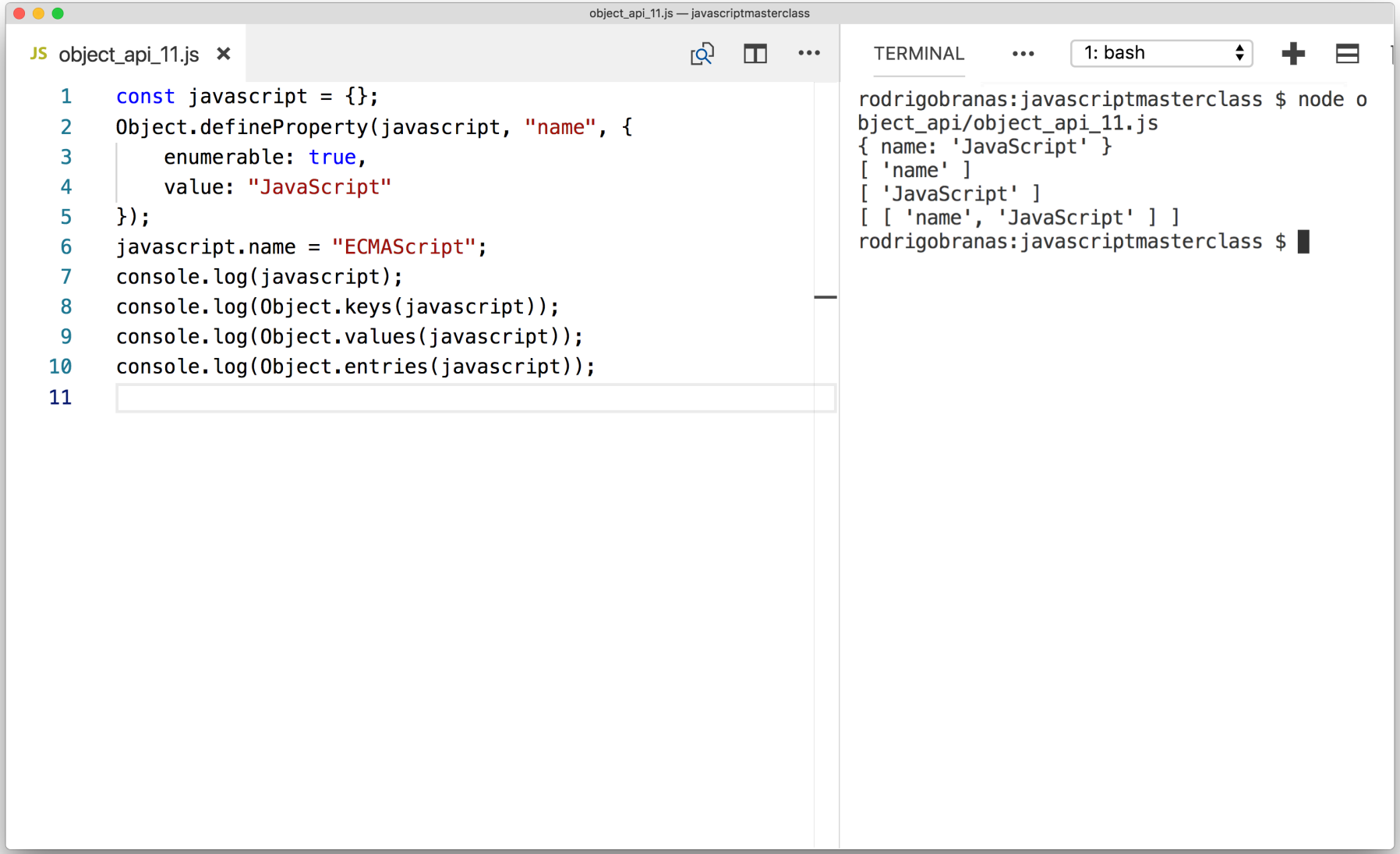

object_api_8.js — javascriptmasterclass
JS object_api_8.js
1 console.log(Object.is(NaN, NaN));
2
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node o
bject_api/object_api_8.js
true
rodrigobranas:javascriptmasterclass $

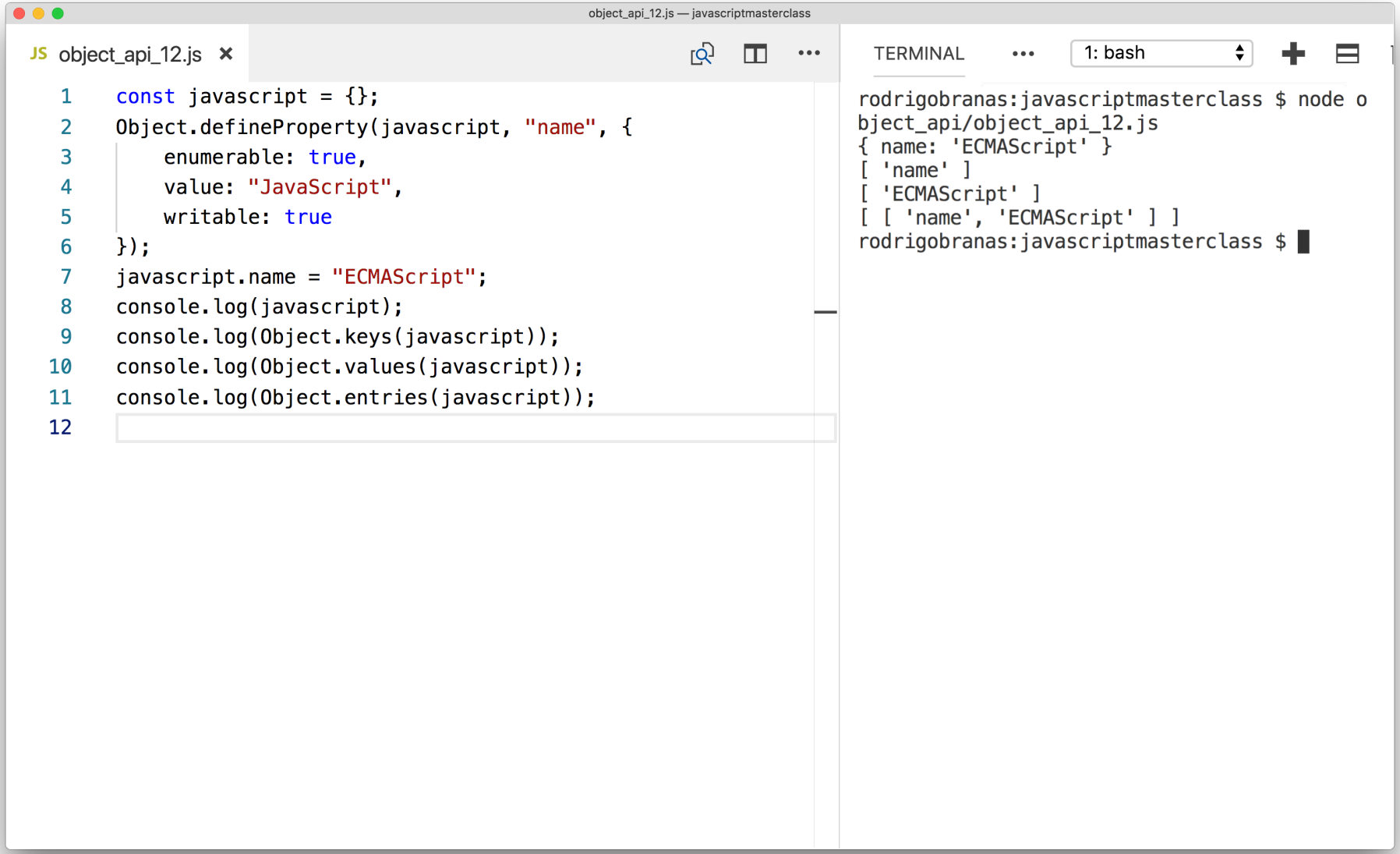
defineProperty

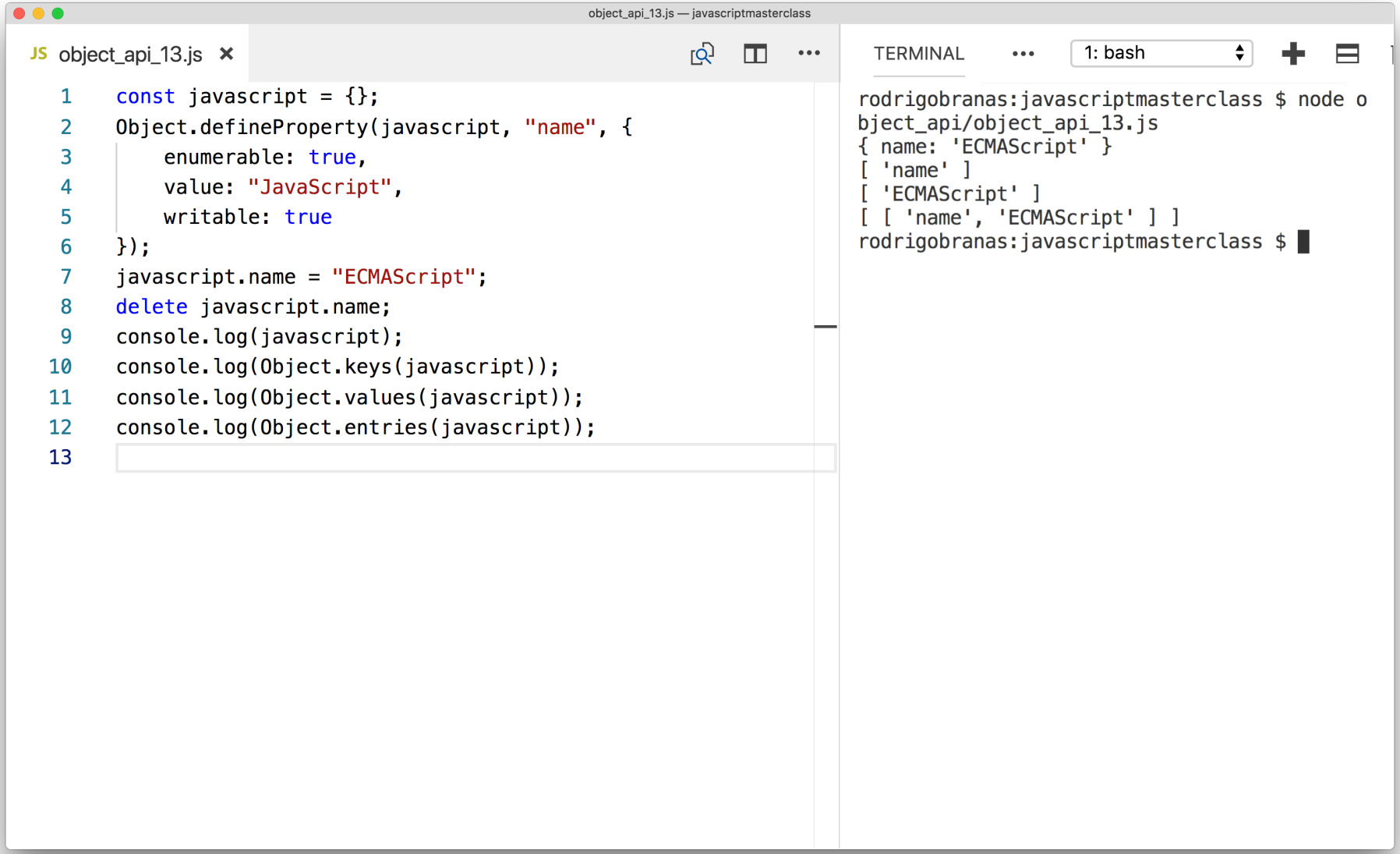
configurable – Permite que uma determinada propriedade seja apagada
enumerable – Permite que uma determinada propriedade seja enumerada
value – Define o valor de uma determinada propriedade
writable – Permite que uma determinada propriedade tenha seu valor modificado

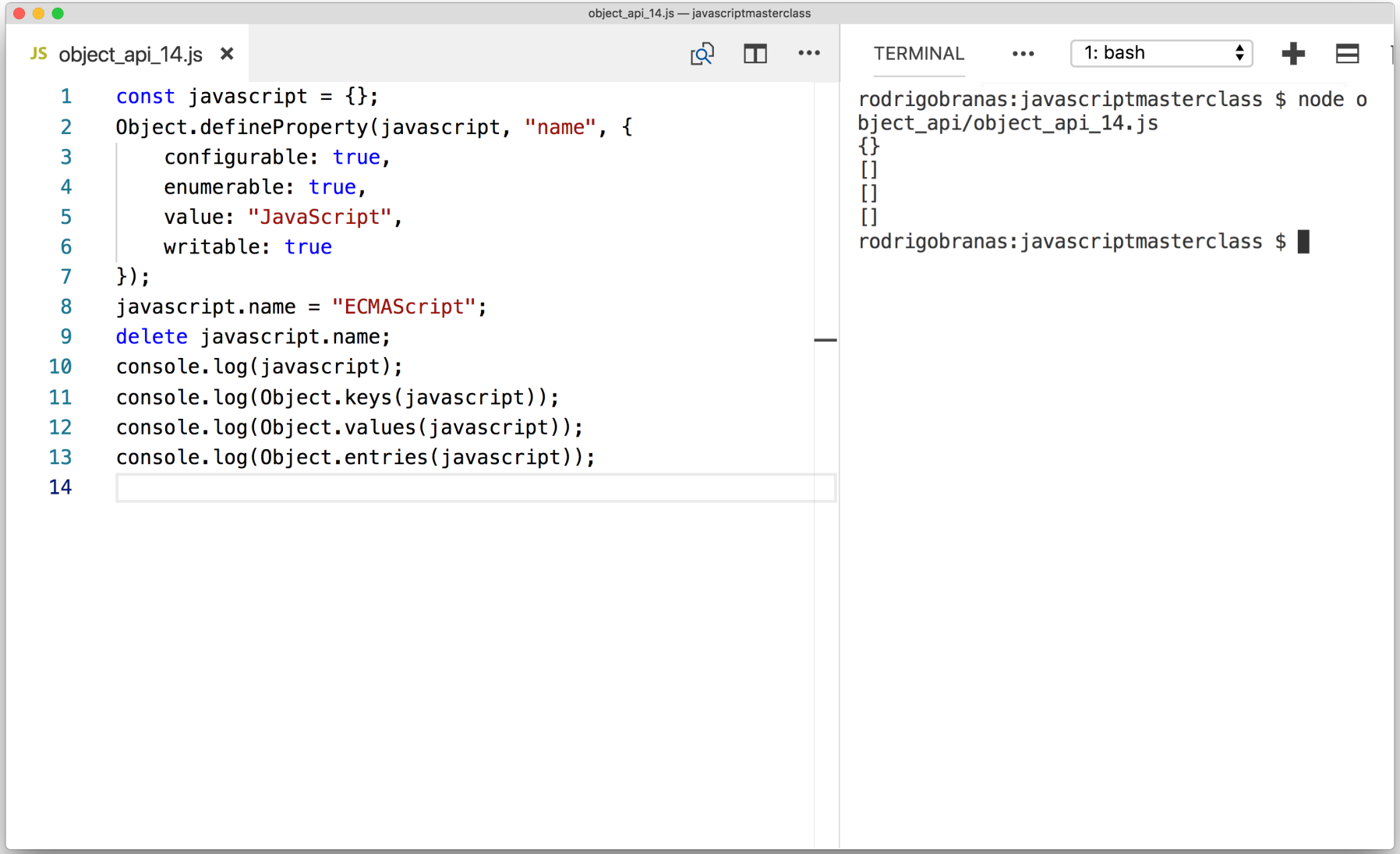

object_api_9.js — javascriptmasterclass
object_api_9.js
1 const javascript = {};
2 Object.defineProperty(javascript, "name", {
3 value: "JavaScript"
4 });
5 console.log(javascript);
6 console.log(Object.keys(javascript));
7 console.log(Object.values(javascript));
8 console.log(Object.entries(javascript));
9
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node o
bject_api/object_api_9.js
{}
[]
[]
[]
rodrigobranas:javascriptmasterclass $


object_api_10.js — javascriptmasterclass
object_api_10.js
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
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node o
bject_api/object_api_10.js
{ name: 'JavaScript' }
['name']
['JavaScript']
[['name', 'JavaScript']]
rodrigobranas:javascriptmasterclass $


object_api_11.js — javascriptmasterclass
object_api_11.js
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
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node o
bject_api/object_api_11.js
{ name: 'JavaScript' }
['name']
['JavaScript']
[['name', 'JavaScript']]
rodrigobranas:javascriptmasterclass $


object_api_12.js — javascriptmasterclass
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
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node o
bject_api/object_api_12.js
{ name: 'ECMAScript' }
['name']
['ECMAScript']
[['name', 'ECMAScript']]
rodrigobranas:javascriptmasterclass $

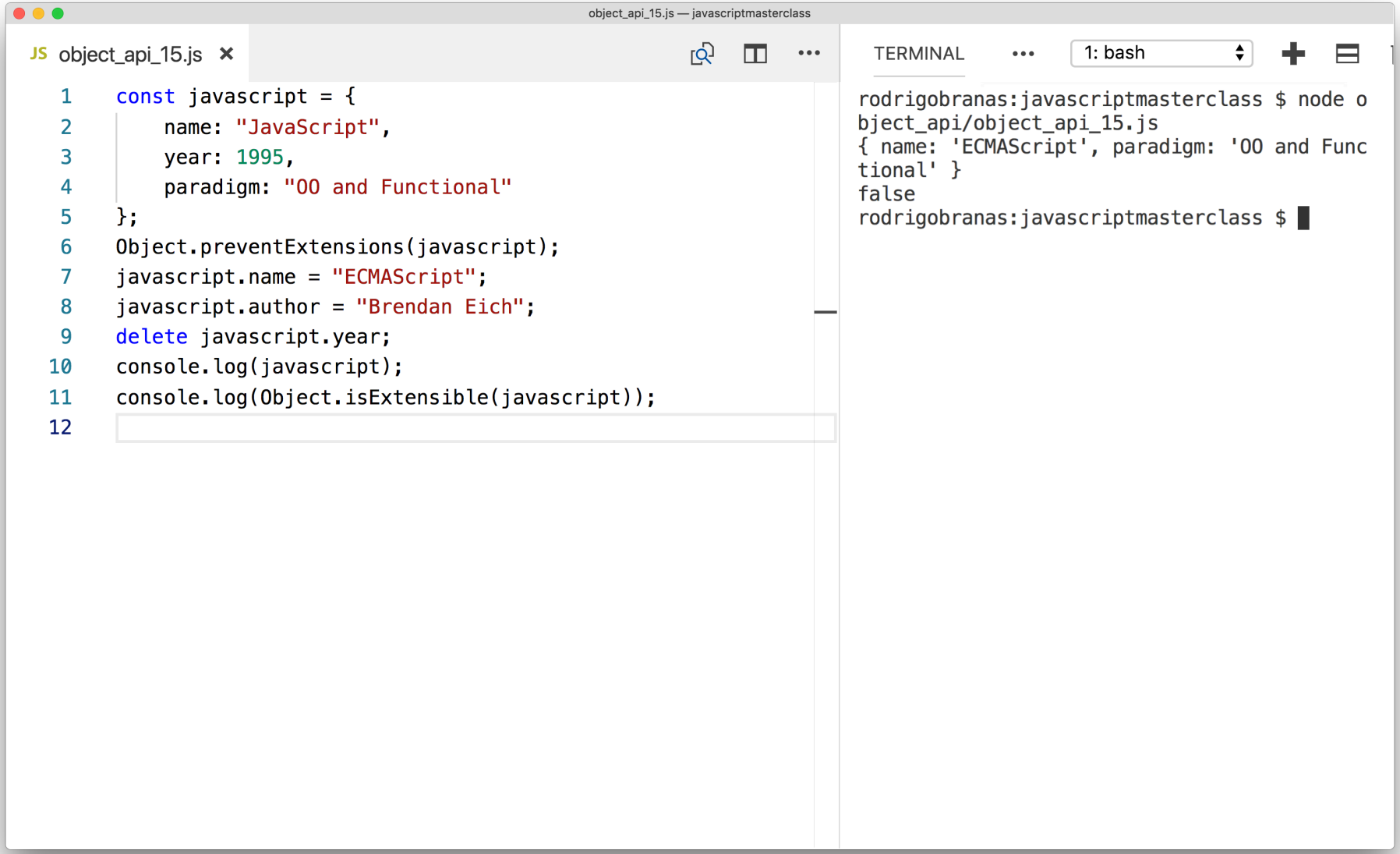

object_api_13.js — javascriptmasterclass
object_api_13.js
1 const javascript = {};
2 Object.defineProperty(javascript, "name", {
3 enumerable: true,
4 value: "JavaScript",
5 writable: true
6 });
7 javascript.name = "ECMAScript";
8 delete javascript.name;
9 console.log(javascript);
10 console.log(Object.keys(javascript));
11 console.log(Object.values(javascript));
12 console.log(Object.entries(javascript));
13
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node o
bject_api/object_api_13.js
{ name: 'ECMAScript' }
['name']
['ECMAScript']
[['name', 'ECMAScript']]
rodrigobranas:javascriptmasterclass $

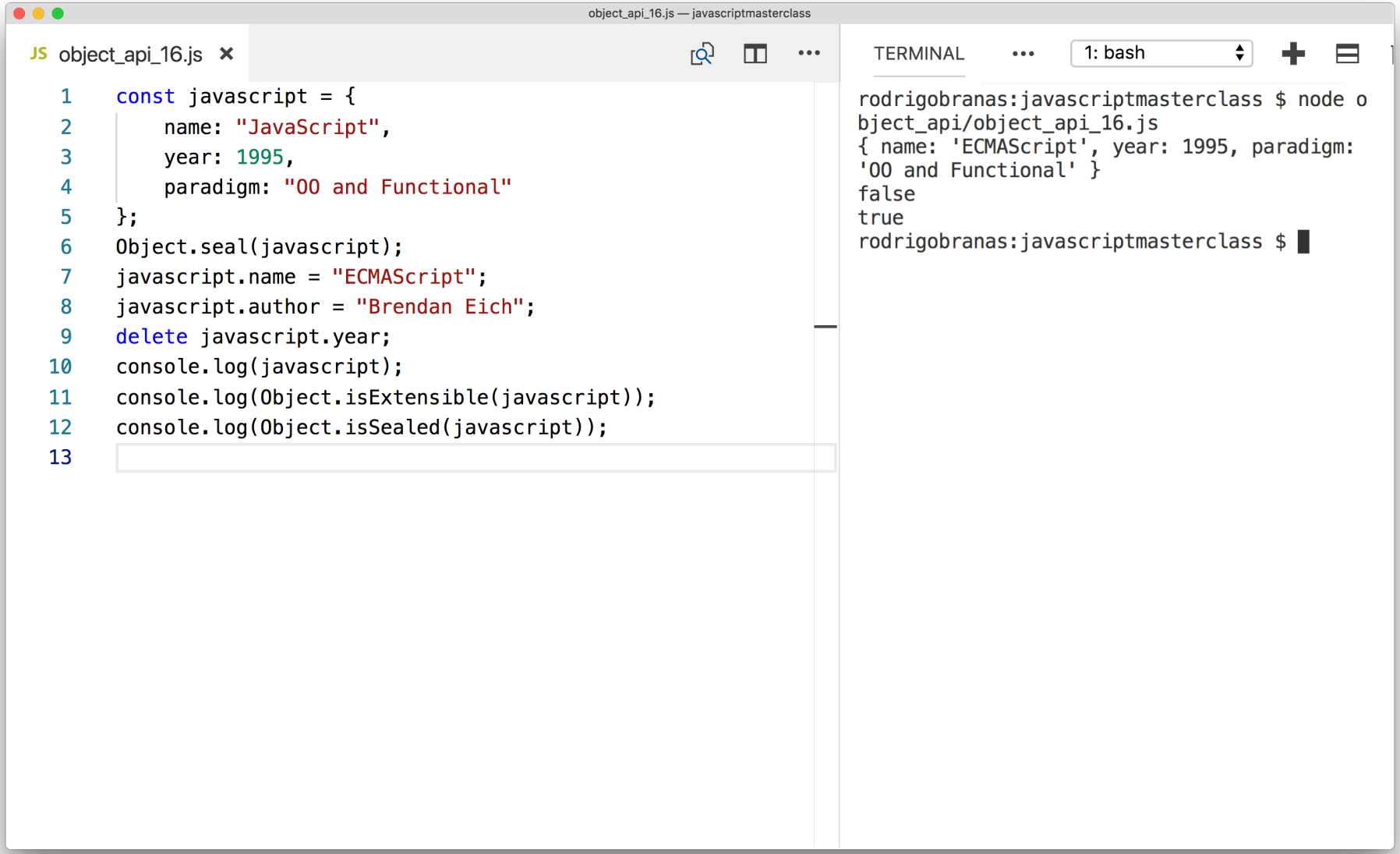

object_api_14.js — javascriptmasterclass
object_api_14.js
1 const javascript = {};
2 Object.defineProperty(javascript, "name", {
3 configurable: true,
4 enumerable: true,
5 value: "JavaScript",
6 writable: true
7 });
8 javascript.name = "ECMAScript";
9 delete javascript.name;
10 console.log(javascript);
11 console.log(Object.keys(javascript));
12 console.log(Object.values(javascript));
13 console.log(Object.entries(javascript));
14
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node o
bject_api/object_api_14.js
{}
[]
[]
[]
rodrigobranas:javascriptmasterclass $

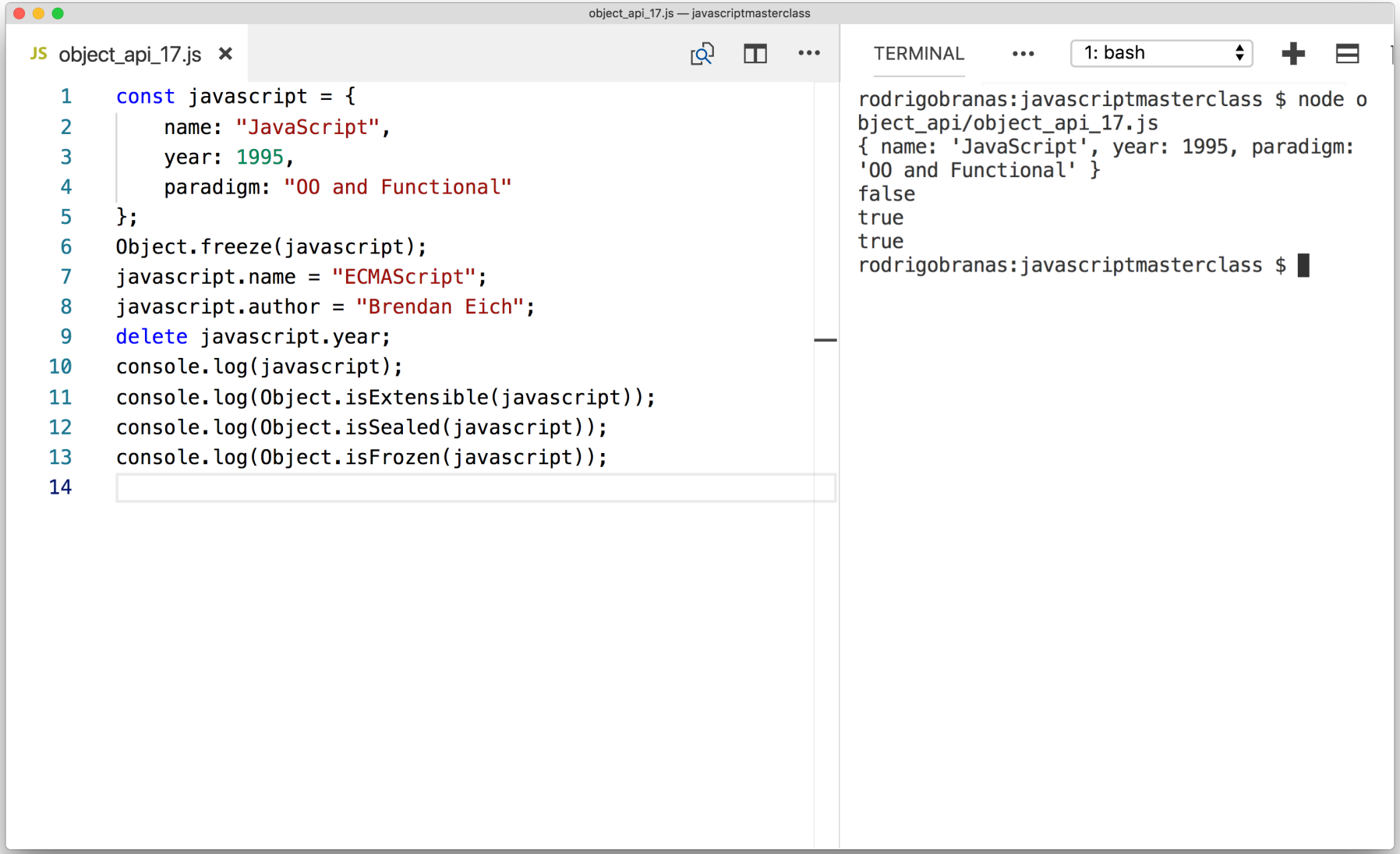
preventExtensions, seal e freeze

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

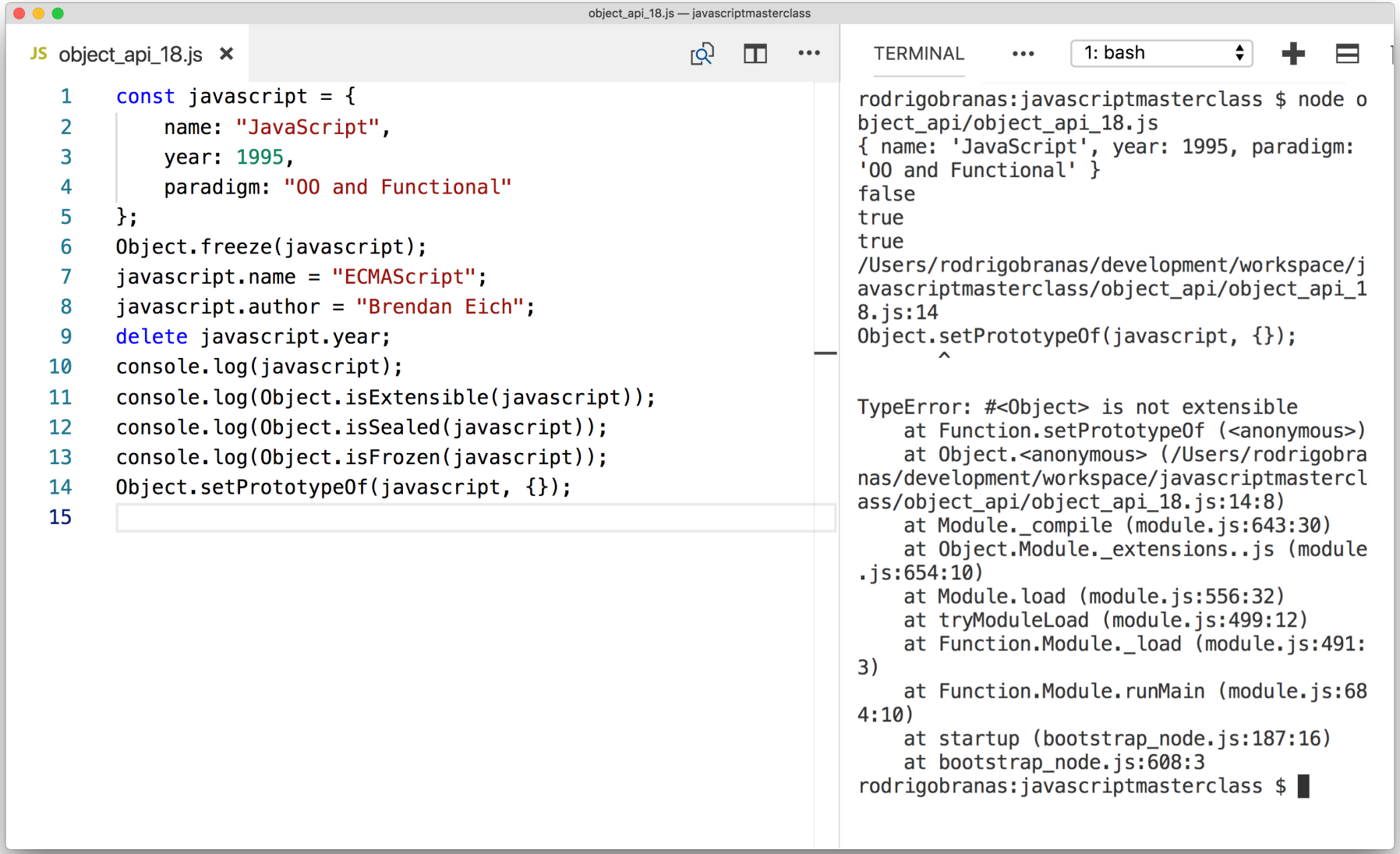
	Create	Read	Update	Delete
Object.preventExtensions	no	yes	yes	yes
Object.seal	no	yes	yes	no
Object.freeze	no	yes	no	no


object_api_15.js — javascriptmasterclass
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
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node o
bject_api/object_api_15.js
{ name: 'ECMAScript', paradigm: 'OO and Func
tional' }
false
rodrigobranas:javascriptmasterclass $


object_api_16.js — javascriptmasterclass
object_api_16.js
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
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node o
bject_api/object_api_16.js
{ name: 'ECMAScript', year: 1995, paradigm:
'OO and Functional' }
false
true
rodrigobranas:javascriptmasterclass $


object_api_17.js — javascriptmasterclass
object_api_17.js
TERMINAL
1: bash
const javascript = {
name: "JavaScript",
year: 1995,
paradigm: "OO and Functional"
};
Object.freeze(javascript);
javascript.name = "ECMAScript";
javascript.author = "Brendan Eich";
delete javascript.year;
console.log(javascript);
console.log(Object.isExtensible(javascript));
console.log(Object.isSealed(javascript));
console.log(Object.isFrozen(javascript));
rodrigobranas:javascriptmasterclass $ node object_api/object_api_17.js
{ name: 'JavaScript', year: 1995, paradigm: 'OO and Functional' }
false
true
true
rodrigobranas:javascriptmasterclass $

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


object_api_18.js — javascriptmasterclass
object_api_18.js
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
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 $