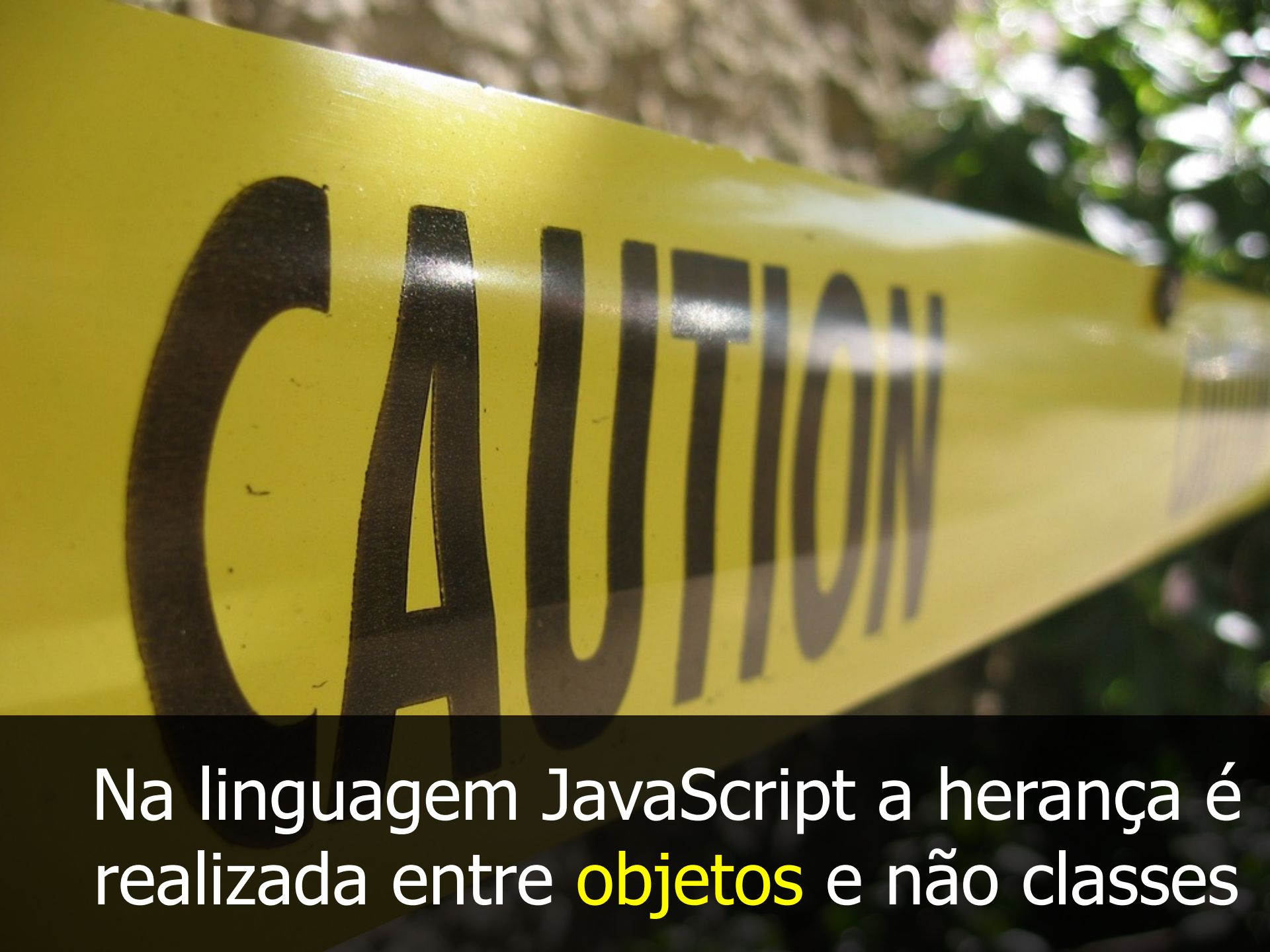
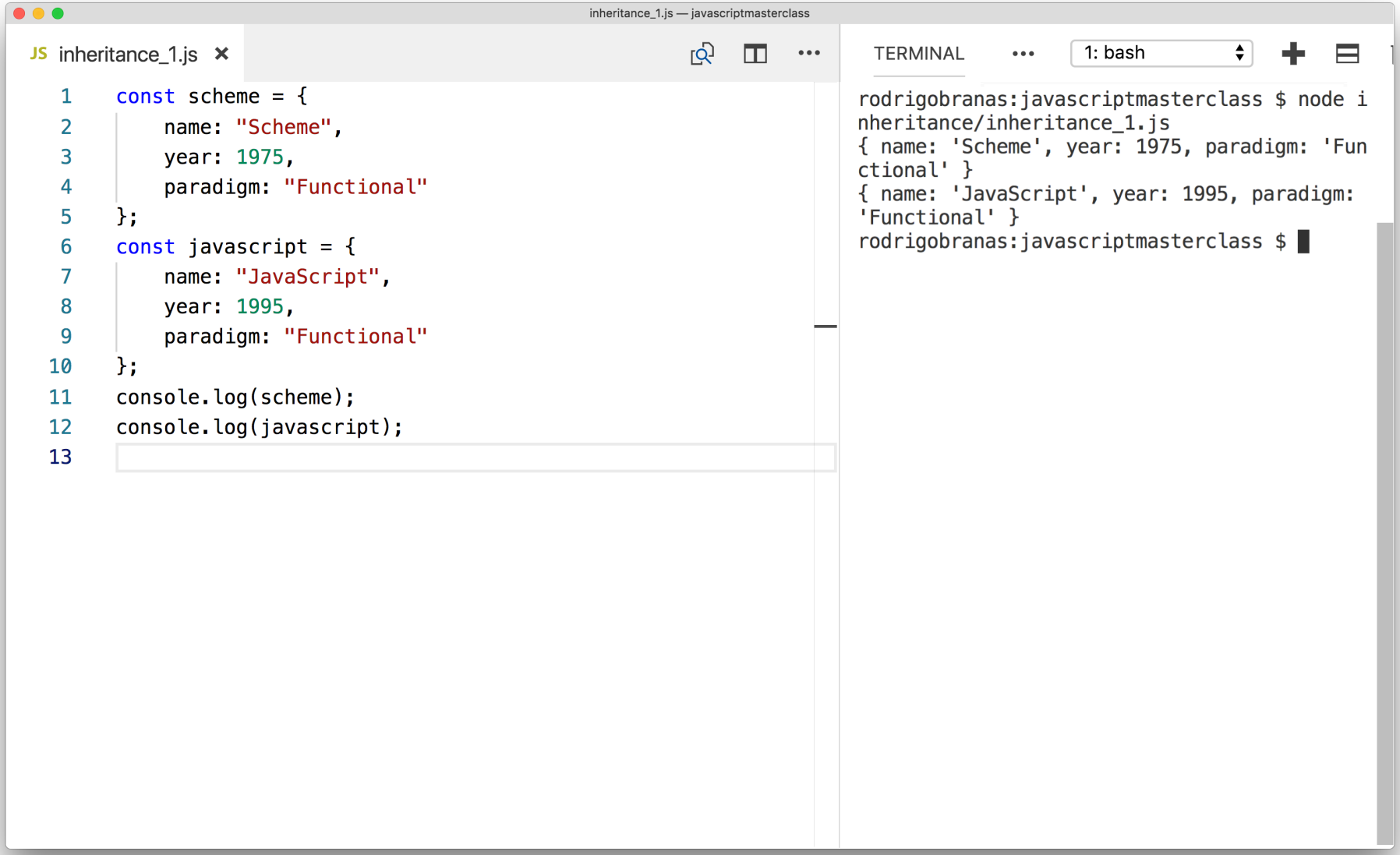

LAST WILL
AND
TESTAMENT
adult residing at
(city).
I revoke all wills and codicils
ARTICLE

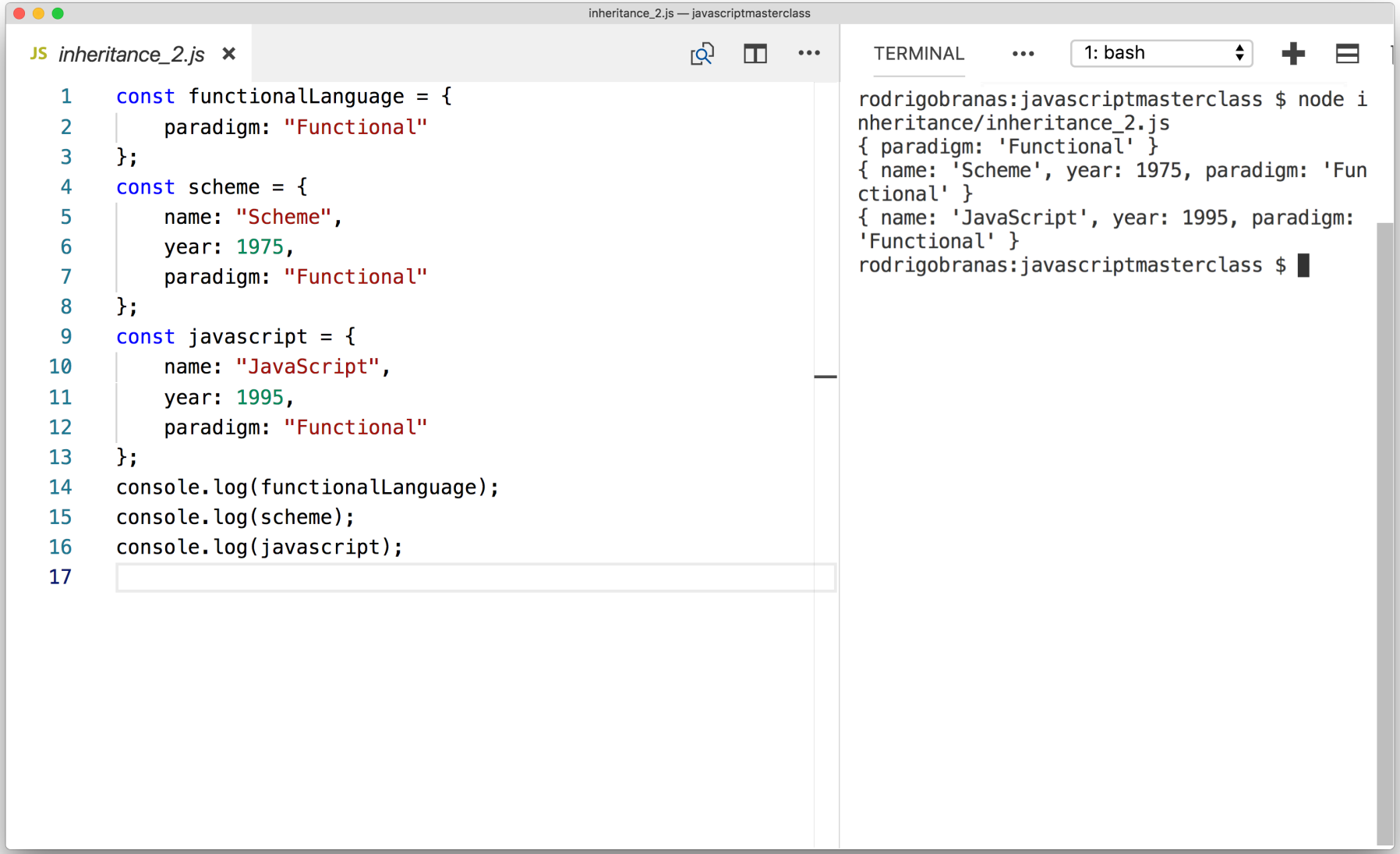
Herança

O principal objetivo da herança é permitir o reuso de código por meio do compartilhamento de propriedades entre objetos, evitando a duplicação

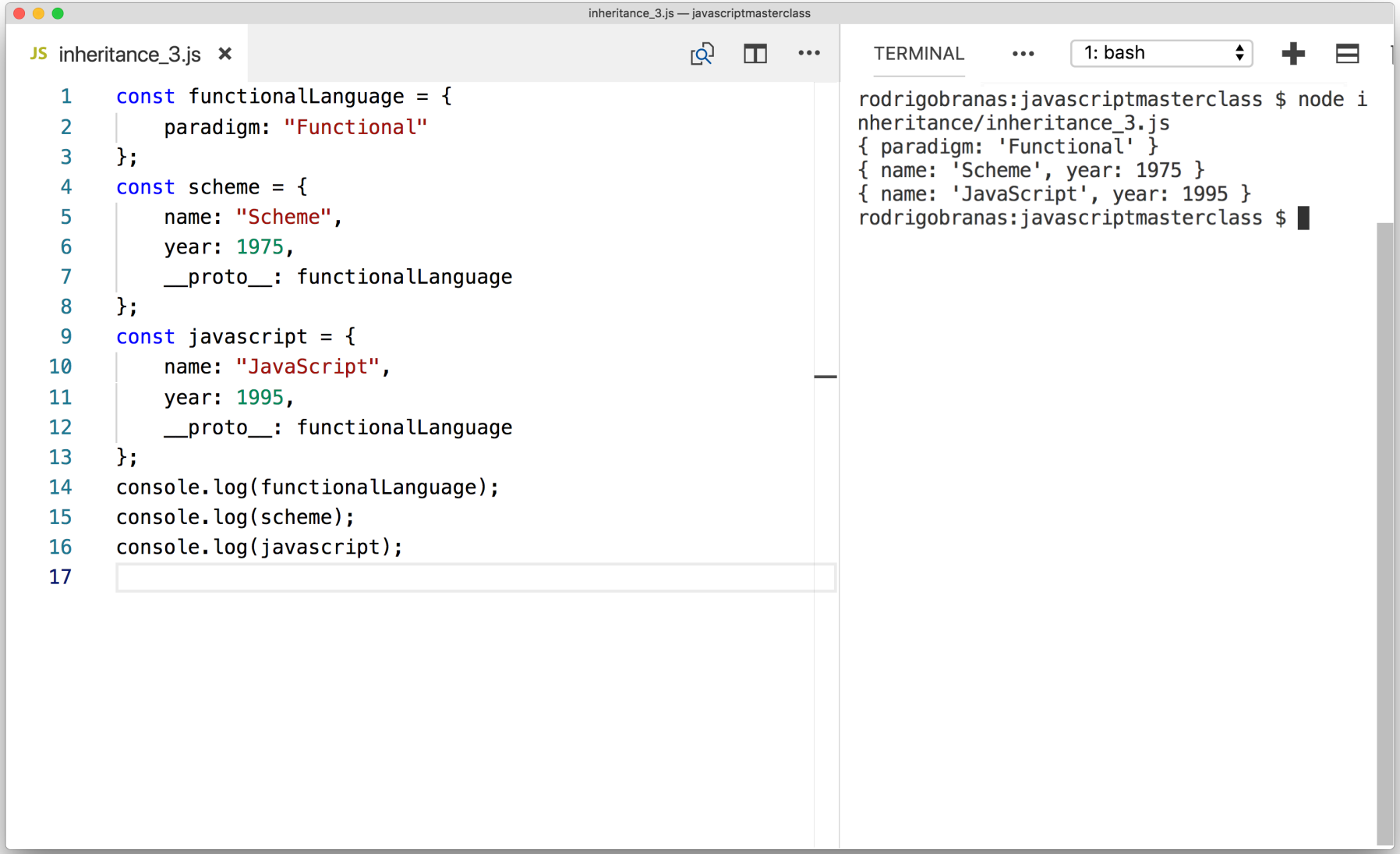


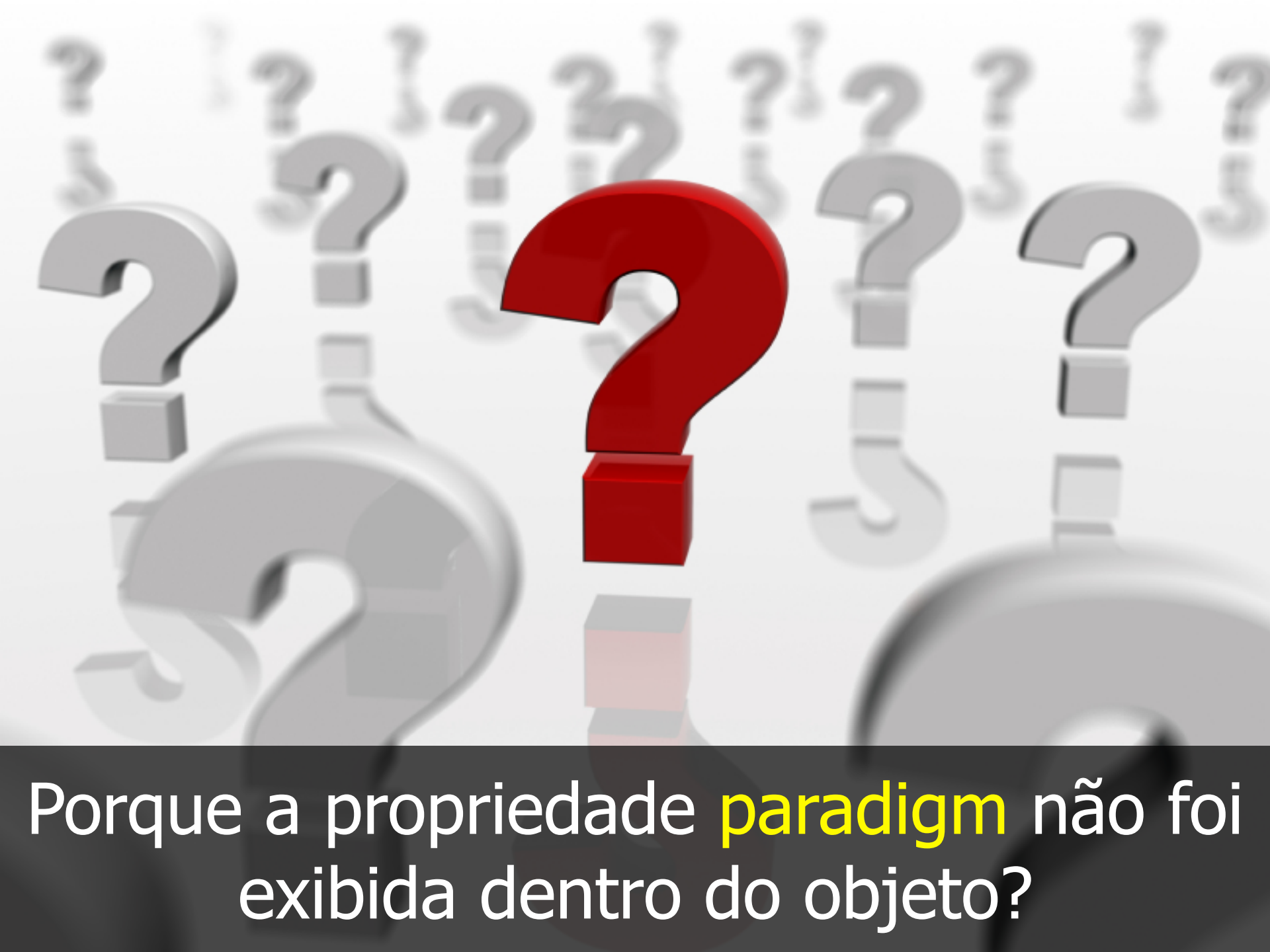
Na linguagem JavaScript a herança é realizada entre objetos e não classes


inheritance_1.js — javascriptmasterclass
JS inheritance_1.js
1 const scheme = {
2 name: "Scheme",
3 year: 1975,
4 paradigm: "Functional"
5 };
6 const javascript = {
7 name: "JavaScript",
8 year: 1995,
9 paradigm: "Functional"
10 };
11 console.log(scheme);
12 console.log(javascript);
13
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node i
nheritance/inheritance_1.js
{ name: 'Scheme', year: 1975, paradigm: 'Fun
ctional' }
{ name: 'JavaScript', year: 1995, paradigm:
'Functional' }
rodrigobranas:javascriptmasterclass $

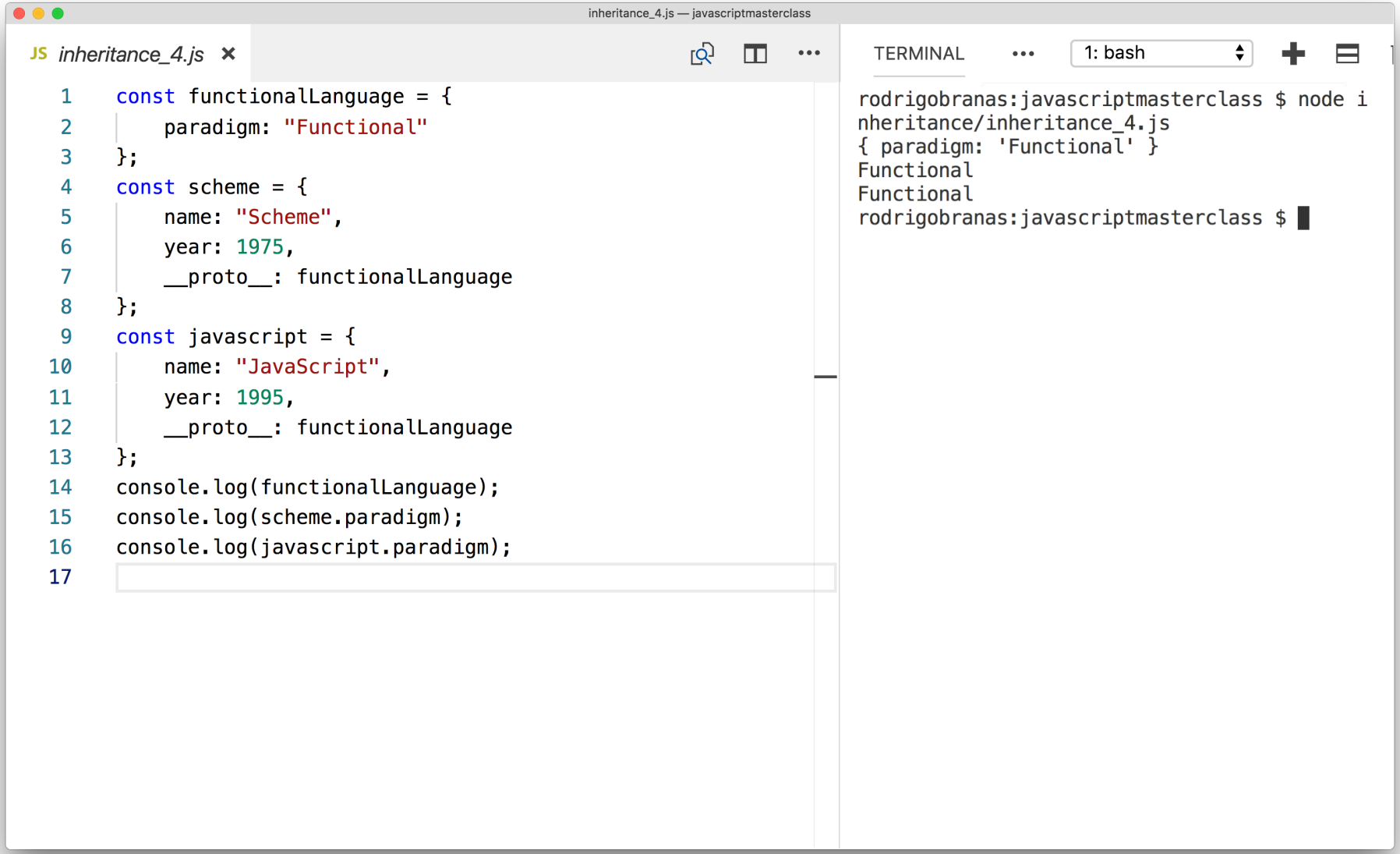

inheritance_2.js — javascriptmasterclass
JS inheritance_2.js
const functionalLanguage = {
paradigm: "Functional"
};
const scheme = {
name: "Scheme",
year: 1975,
paradigm: "Functional"
};
const javascript = {
name: "JavaScript",
year: 1995,
paradigm: "Functional"
};
console.log(functionalLanguage);
console.log(scheme);
console.log(javascript);
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node inheritance/inheritance_2.js
{ paradigm: 'Functional' }
{ name: 'Scheme', year: 1975, paradigm: 'Functional' }
{ name: 'JavaScript', year: 1995, paradigm: 'Functional' }
rodrigobranas:javascriptmasterclass $

A propriedade __proto__ é uma referência para o protótipo do objeto

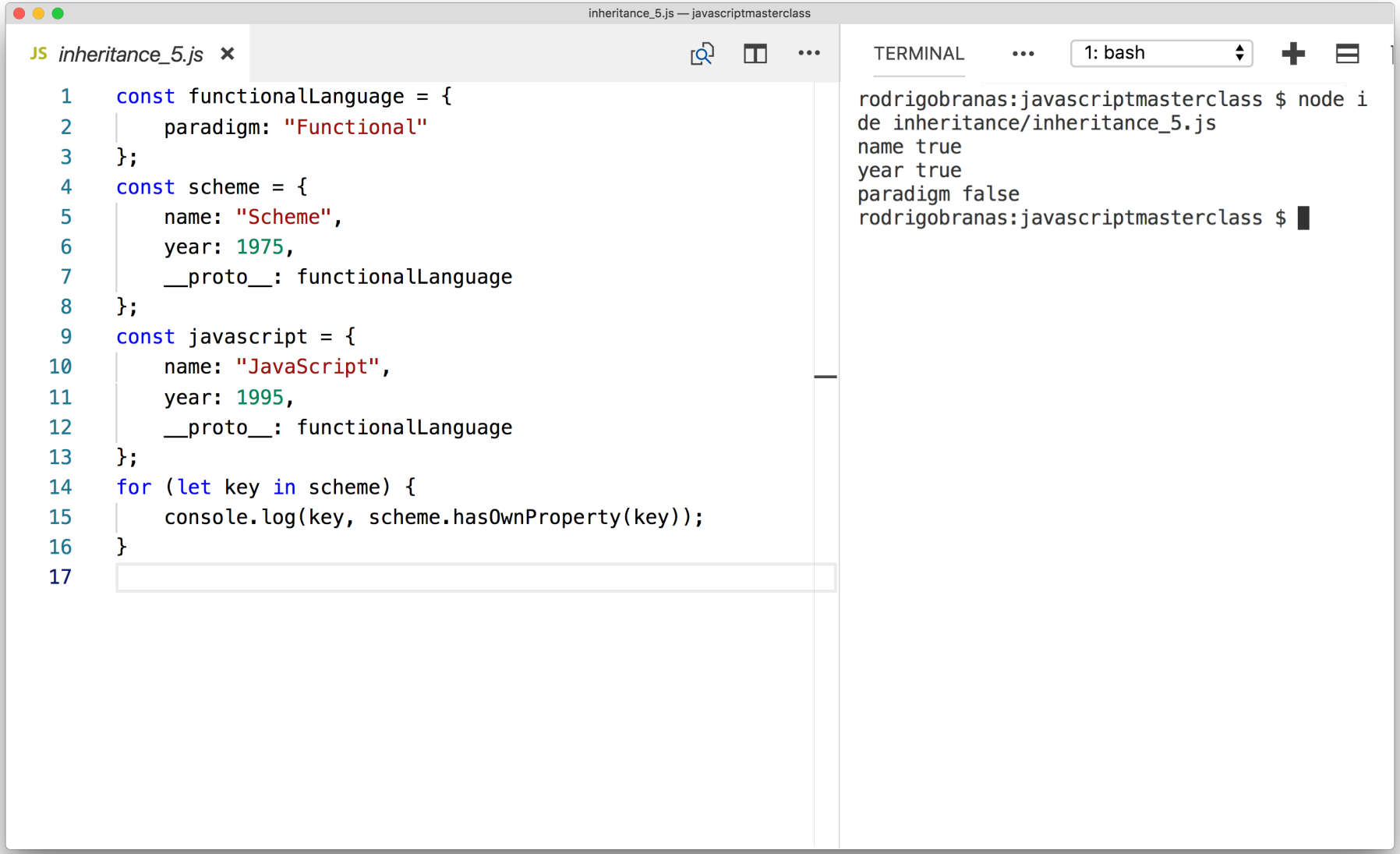

inheritance_3.js — javascriptmasterclass
JS inheritance_3.js
1 const functionalLanguage = {
2 paradigm: "Functional"
3 };
4 const scheme = {
5 name: "Scheme",
6 year: 1975,
7 __proto__: functionalLanguage
8 };
9 const javascript = {
10 name: "JavaScript",
11 year: 1995,
12 __proto__: functionalLanguage
13 };
14 console.log(functionalLanguage);
15 console.log(scheme);
16 console.log(javascript);
17
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node i
nheritance/inheritance_3.js
{ paradigm: 'Functional' }
{ name: 'Scheme', year: 1975 }
{ name: 'JavaScript', year: 1995 }
rodrigobranas:javascriptmasterclass $



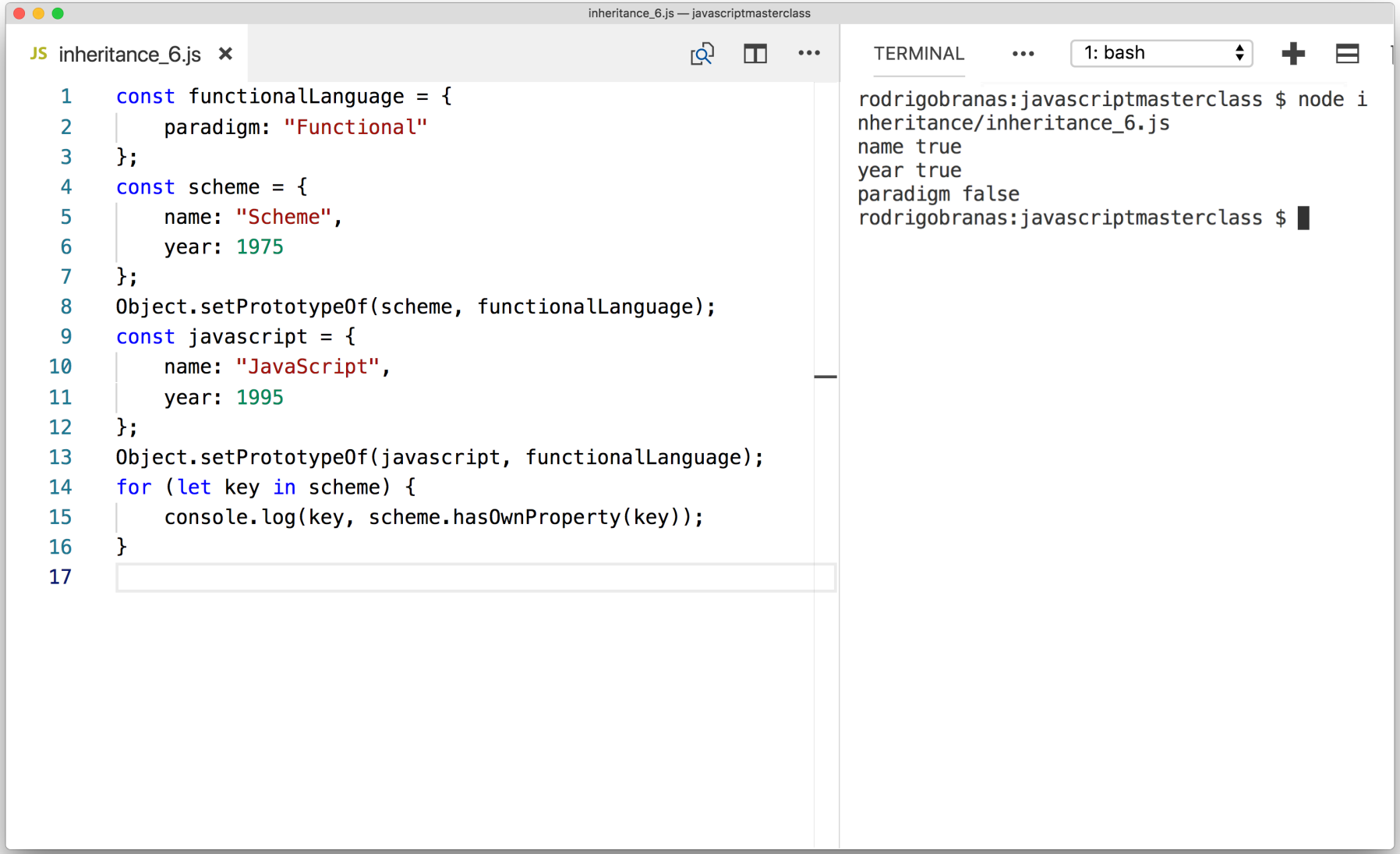
Porque a propriedade paradigm não foi exibida dentro do objeto?


inheritance_4.js — javascriptmasterclass
JS inheritance_4.js
1 const functionalLanguage = {
2 paradigm: "Functional"
3 };
4 const scheme = {
5 name: "Scheme",
6 year: 1975,
7 __proto__: functionalLanguage
8 };
9 const javascript = {
10 name: "JavaScript",
11 year: 1995,
12 __proto__: functionalLanguage
13 };
14 console.log(functionalLanguage);
15 console.log(scheme.paradigm);
16 console.log(javascript.paradigm);
17
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node i
nheritance/inheritance_4.js
{ paradigm: 'Functional' }
Functional
Functional
rodrigobranas:javascriptmasterclass $

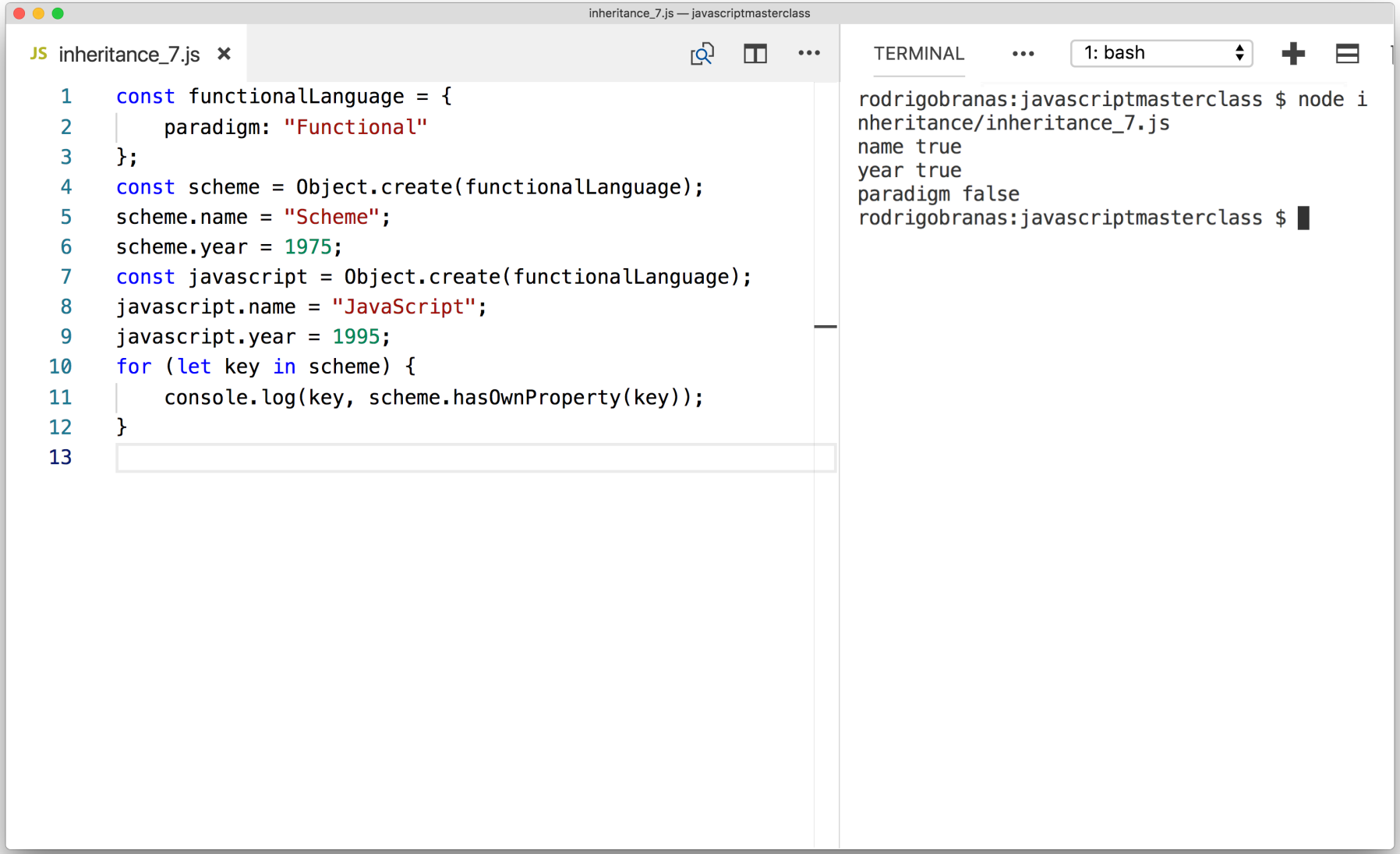
O método hasOwnProperty pode ser utilizado para determinar se uma propriedade pertence ao objeto

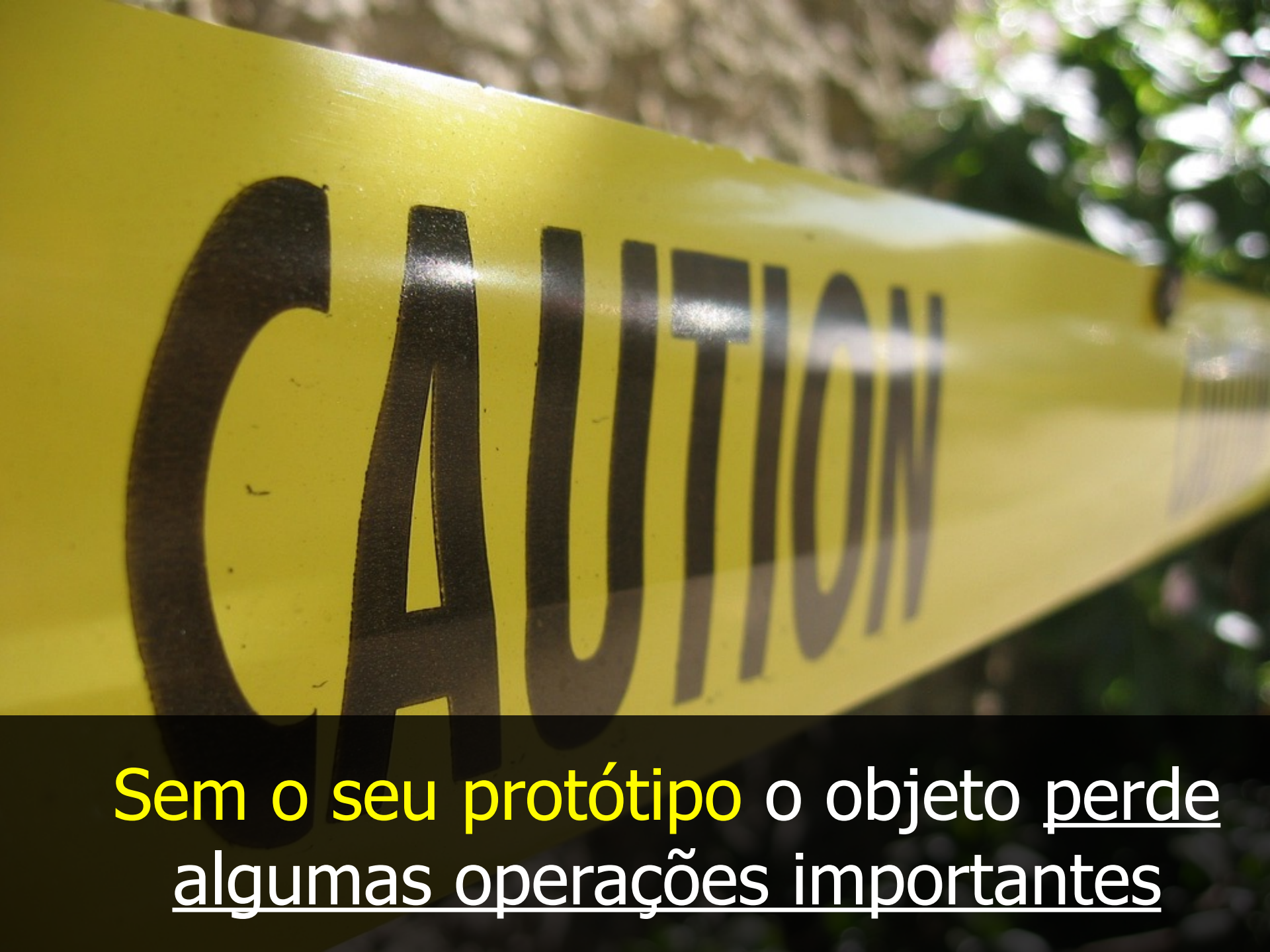

inheritance_5.js — javascriptmasterclass
JS inheritance_5.js
const functionalLanguage = {
paradigm: "Functional"
};
const scheme = {
name: "Scheme",
year: 1975,
__proto__: functionalLanguage
};
const javascript = {
name: "JavaScript",
year: 1995,
__proto__: functionalLanguage
};
for (let key in scheme) {
console.log(key, scheme.hasOwnProperty(key));
}
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node i
de inheritance/inheritance_5.js
name true
year true
paradigm false
rodrigobranas:javascriptmasterclass $

Os métodos Object.setPrototypeOf e Object.getPrototypeOf permitem a interação com o protótipo do objeto

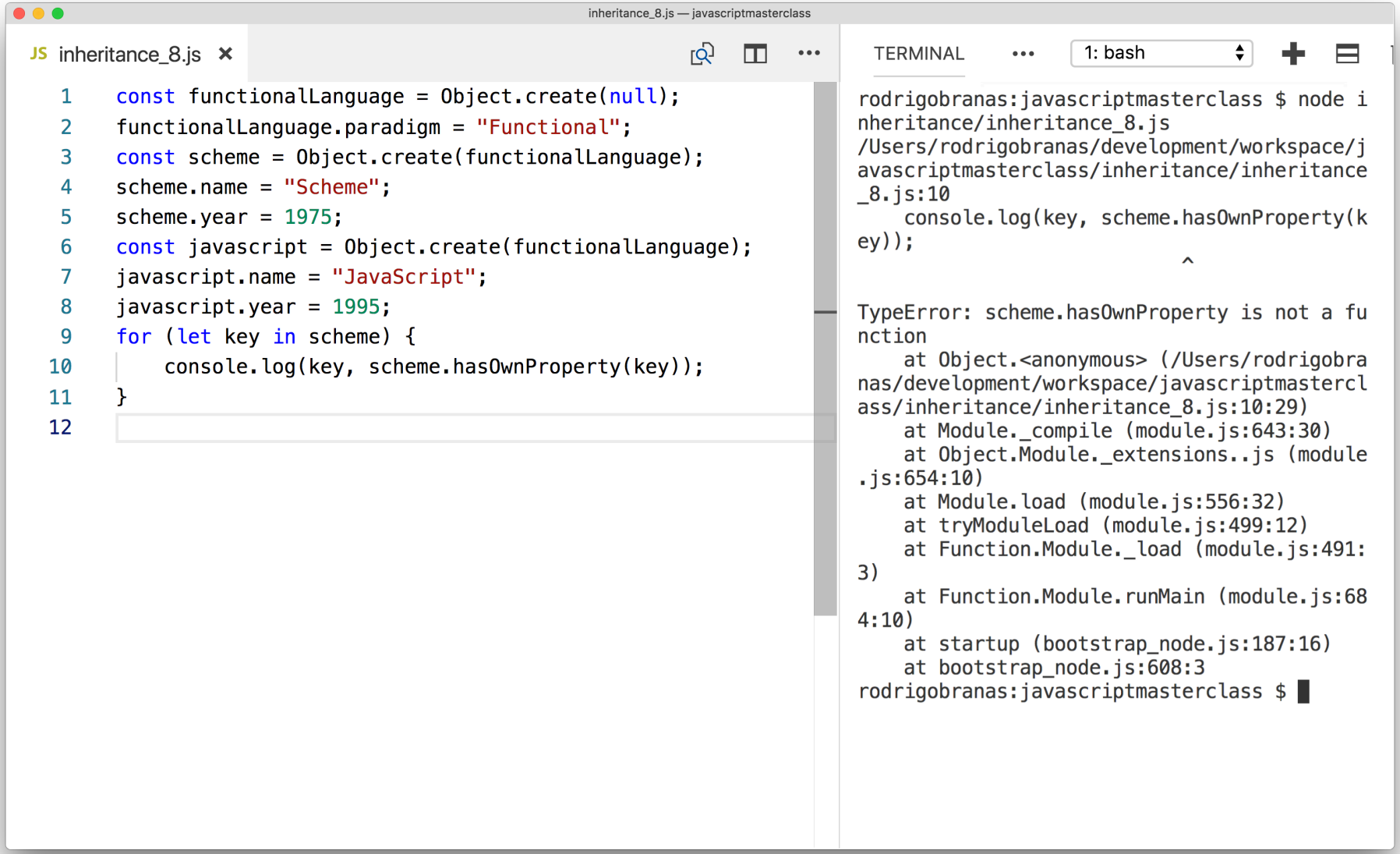

inheritance_6.js — javascriptmasterclass
inheritance_6.js
1 const functionalLanguage = {
2 paradigm: "Functional"
3 };
4 const scheme = {
5 name: "Scheme",
6 year: 1975
7 };
8 Object.setPrototypeOf(scheme, functionalLanguage);
9 const javascript = {
10 name: "JavaScript",
11 year: 1995
12 };
13 Object.setPrototypeOf(javascript, functionalLanguage);
14 for (let key in scheme) {
15 console.log(key, scheme.hasOwnProperty(key));
16 }
17
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node i
nheritance/inheritance_6.js
name true
year true
paradigm false
rodrigobranas:javascriptmasterclass $

Com o método Object.create é possível criar um objeto passando o seu protótipo por parâmetro

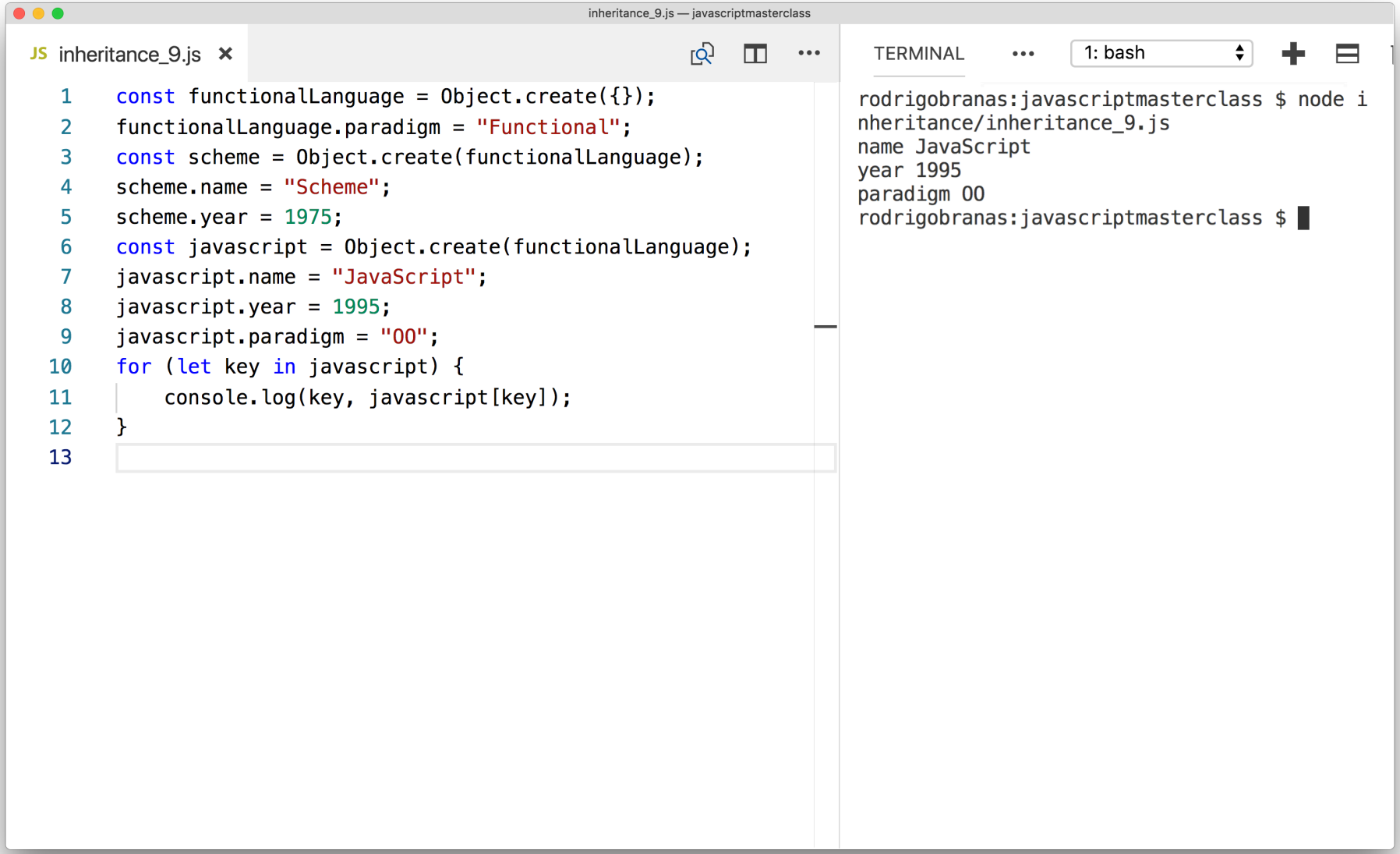

inheritance_7.js — javascriptmasterclass
JS inheritance_7.js
1 const functionalLanguage = {
2 paradigm: "Functional"
3 };
4 const scheme = Object.create(functionalLanguage);
5 scheme.name = "Scheme";
6 scheme.year = 1975;
7 const javascript = Object.create(functionalLanguage);
8 javascript.name = "JavaScript";
9 javascript.year = 1995;
10 for (let key in scheme) {
11 console.log(key, scheme.hasOwnProperty(key));
12 }
13
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node i
nheritance/inheritance_7.js
name true
year true
paradigm false
rodrigobranas:javascriptmasterclass $

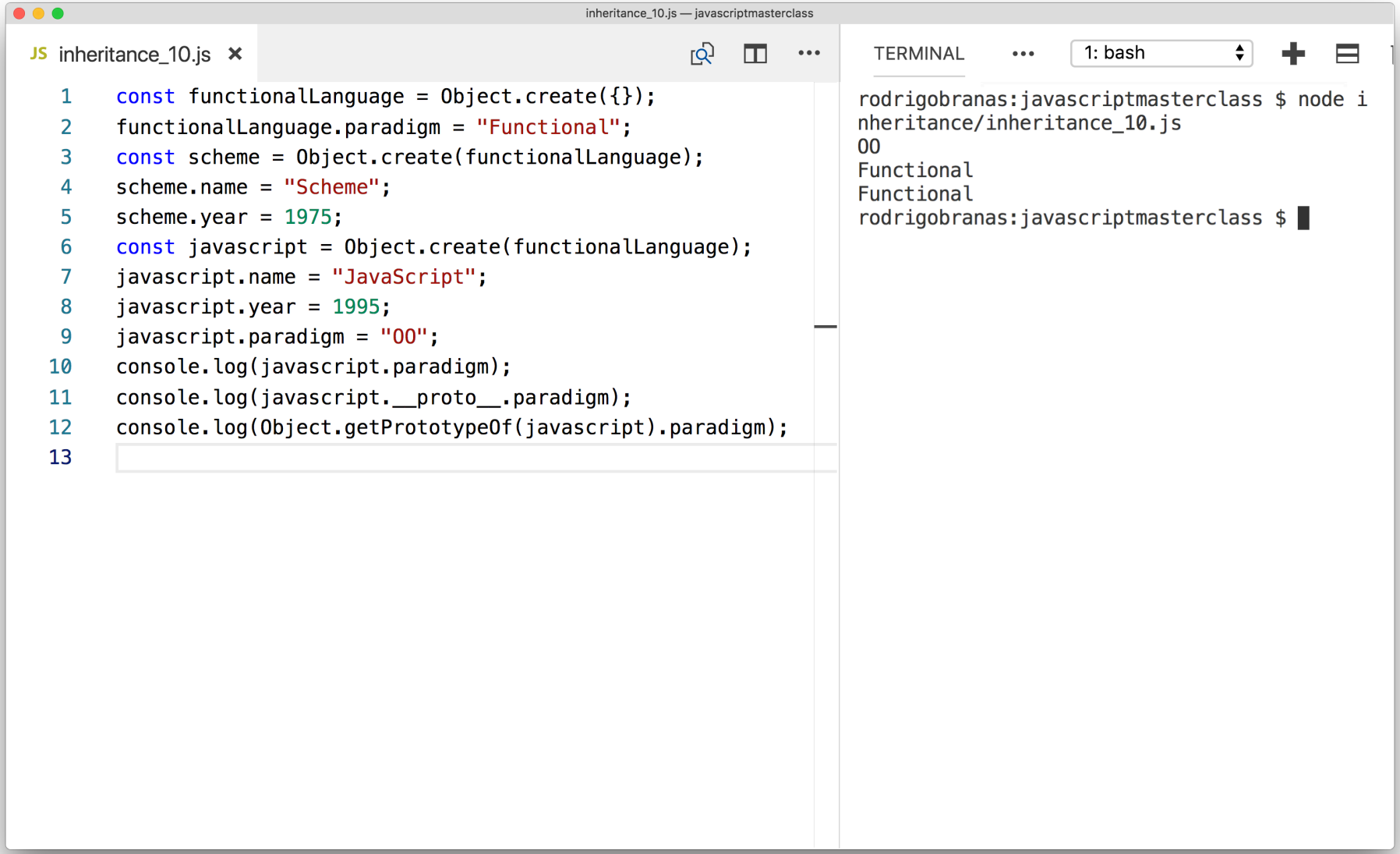


Sem o seu protótipo o objeto perde algumas operações importantes


inheritance_8.js — javascriptmasterclass
inheritance_8.js
1 const functionalLanguage = Object.create(null);
2 functionalLanguage.paradigm = "Functional";
3 const scheme = Object.create(functionalLanguage);
4 scheme.name = "Scheme";
5 scheme.year = 1975;
6 const javascript = Object.create(functionalLanguage);
7 javascript.name = "JavaScript";
8 javascript.year = 1995;
9 for (let key in scheme) {
10 console.log(key, scheme.hasOwnProperty(key));
11 }
12
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node inheritance/inheritance_8.js
/Users/rodrigobranas/development/workspace/javascriptmasterclass/inheritance/inheritance_8.js:10
console.log(key, scheme.hasOwnProperty(key));
^
TypeError: scheme.hasOwnProperty is not a function
at Object.<anonymous> (/Users/rodrigobranas/development/workspace/javascriptmasterclass/inheritance/inheritance_8.js:10:29)
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 $

Caso a mesma propriedade exista no objeto e no seu protótipo, a propriedade do próprio objeto é retornada, fazendo sombra à propriedade do protótipo


inheritance_9.js — javascriptmasterclass
JS inheritance_9.js
1 const functionalLanguage = Object.create({});
2 functionalLanguage.paradigm = "Functional";
3 const scheme = Object.create(functionalLanguage);
4 scheme.name = "Scheme";
5 scheme.year = 1975;
6 const javascript = Object.create(functionalLanguage);
7 javascript.name = "JavaScript";
8 javascript.year = 1995;
9 javascript.paradigm = "OO";
10 for (let key in javascript) {
11 console.log(key, javascript[key]);
12 }
13
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node i
nheritance/inheritance_9.js
name JavaScript
year 1995
paradigm OO
rodrigobranas:javascriptmasterclass $


inheritance_10.js — javascriptmasterclass
inheritance_10.js
1 const functionalLanguage = Object.create({});
2 functionalLanguage.paradigm = "Functional";
3 const scheme = Object.create(functionalLanguage);
4 scheme.name = "Scheme";
5 scheme.year = 1975;
6 const javascript = Object.create(functionalLanguage);
7 javascript.name = "JavaScript";
8 javascript.year = 1995;
9 javascript.paradigm = "OO";
10 console.log(javascript.paradigm);
11 console.log(javascript.__proto__.paradigm);
12 console.log(Object.getPrototypeOf(javascript).paradigm);
13
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node i
nheritance/inheritance_10.js
OO
Functional
Functional
rodrigobranas:javascriptmasterclass $