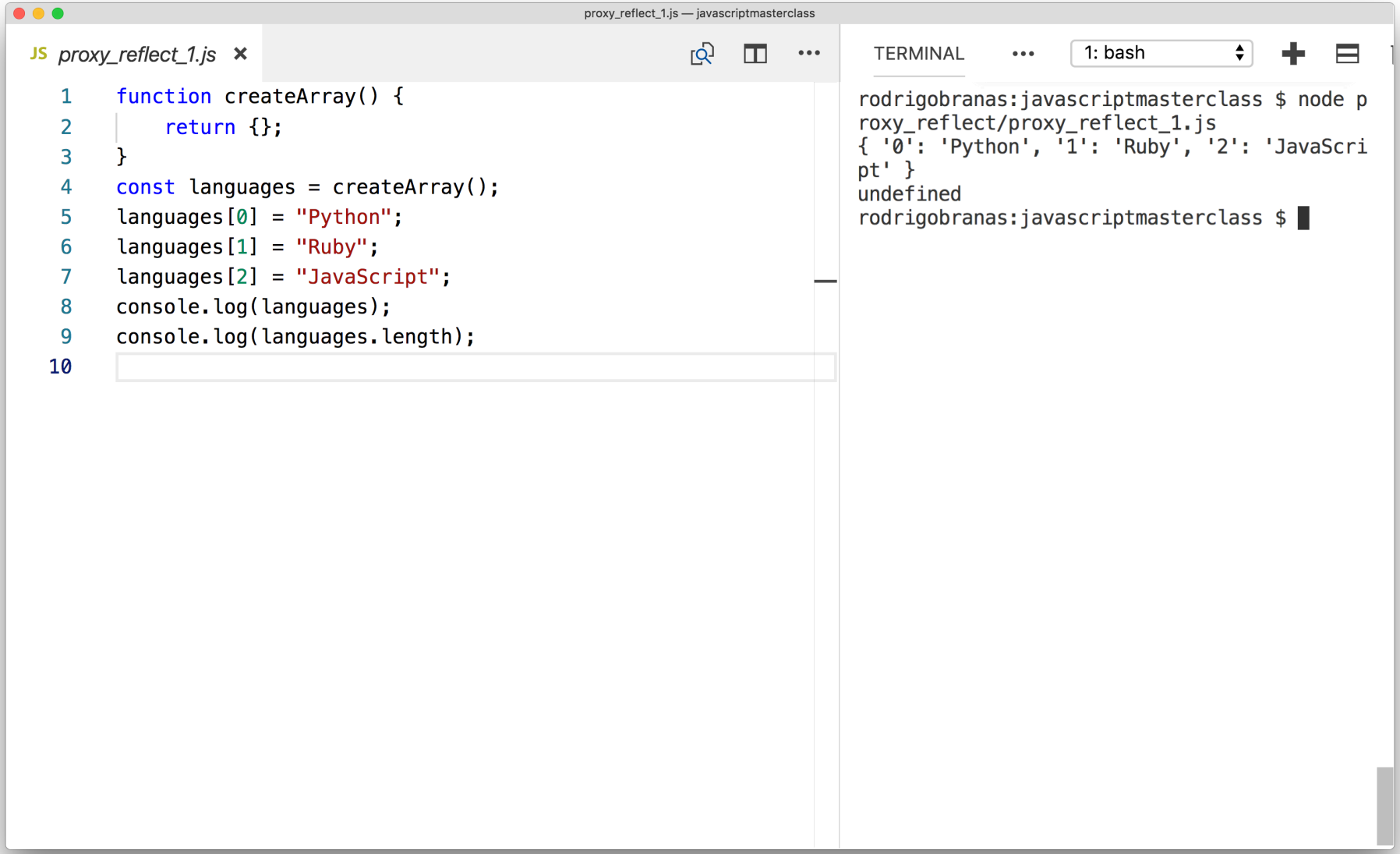
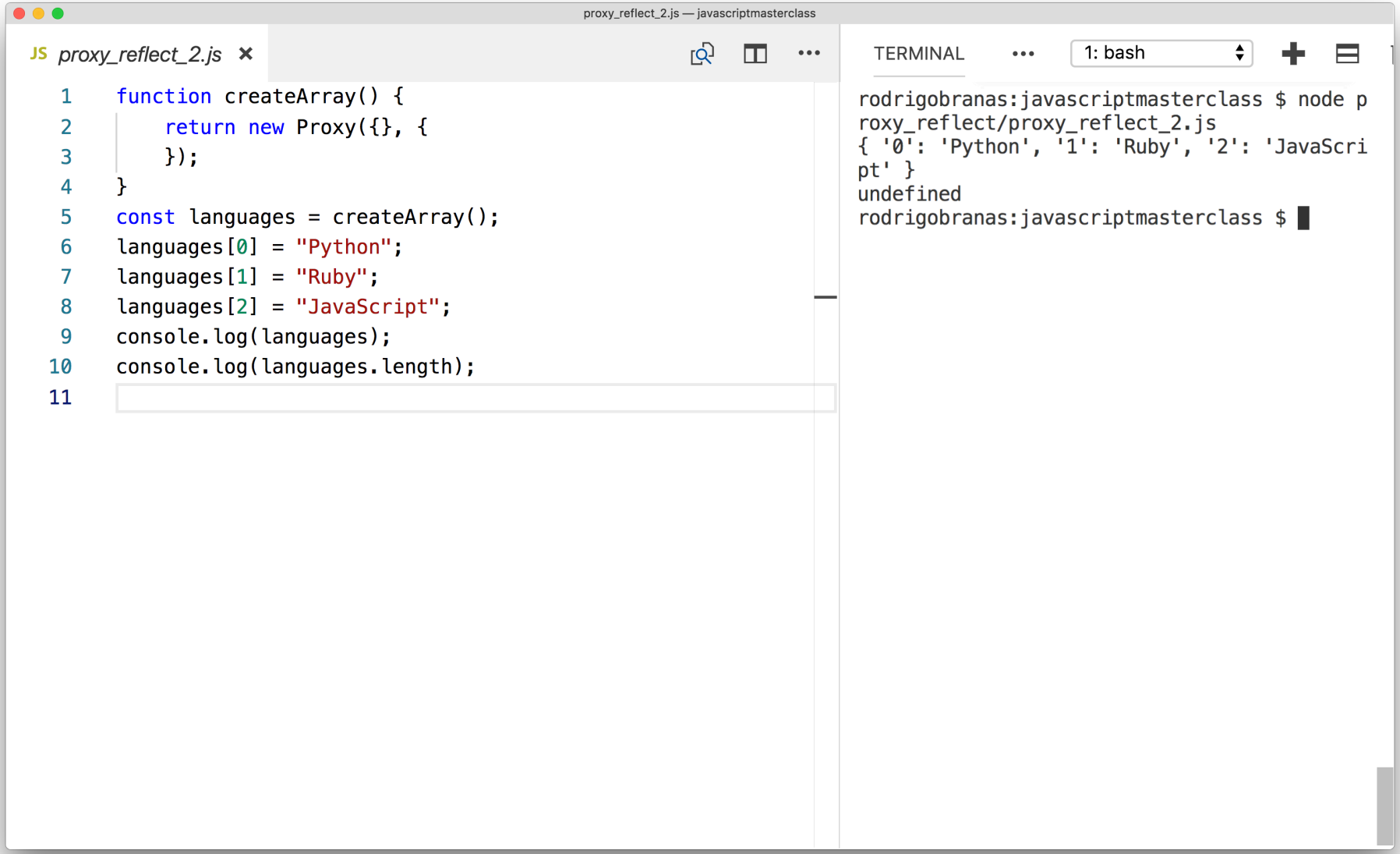
Proxy e Reflect

Um proxy é capaz de interceptar diversos tipos de operações em um objeto alvo

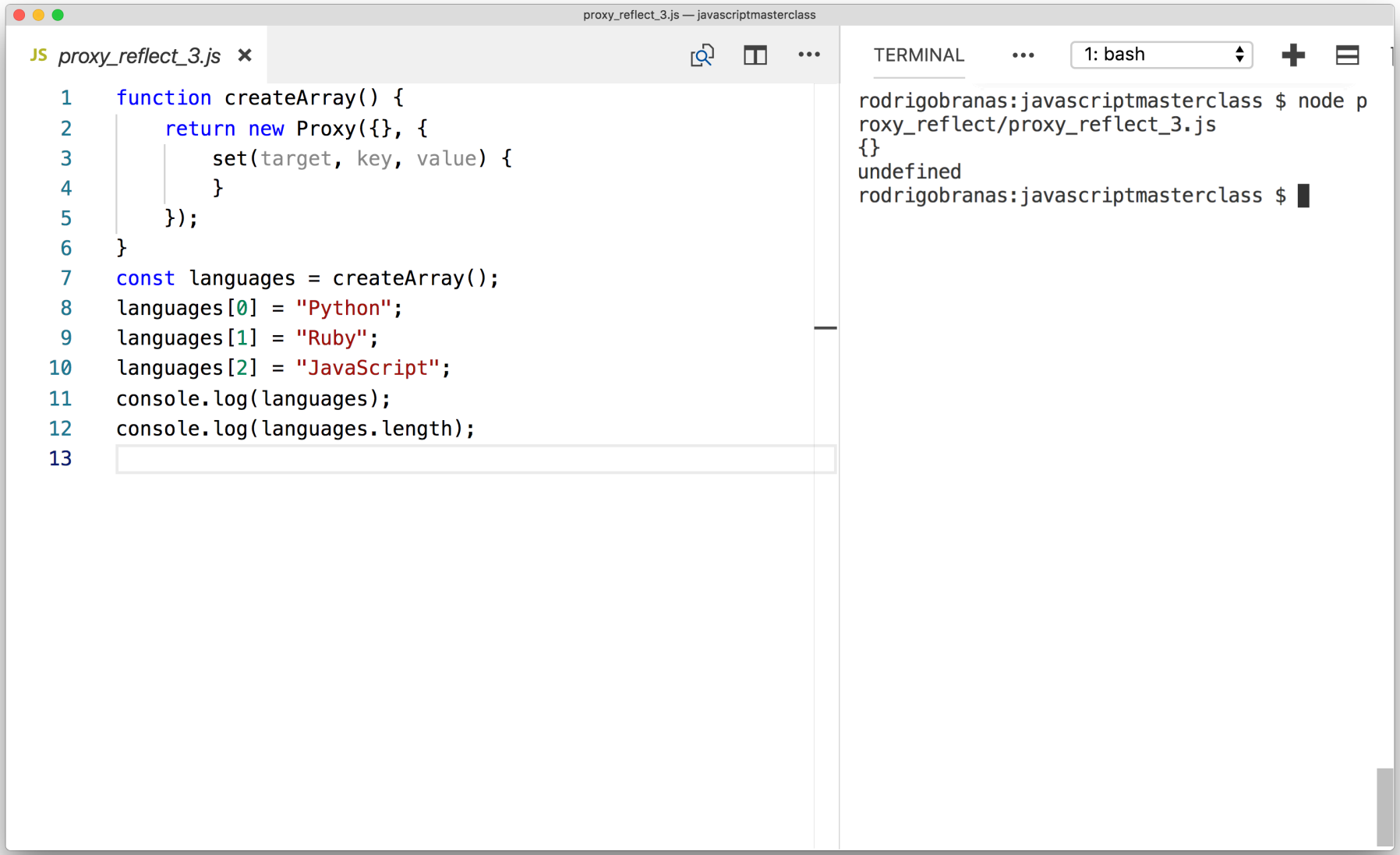

proxy_reflect_1.js — javascriptmasterclass
JS proxy_reflect_1.js
function createArray() {
return {};
}
const languages = createArray();
languages[0] = "Python";
languages[1] = "Ruby";
languages[2] = "JavaScript";
console.log(languages);
console.log(languages.length);
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node proxy_reflect/proxy_reflect_1.js
{ '0': 'Python', '1': 'Ruby', '2': 'JavaScript' }
undefined
rodrigobranas:javascriptmasterclass $

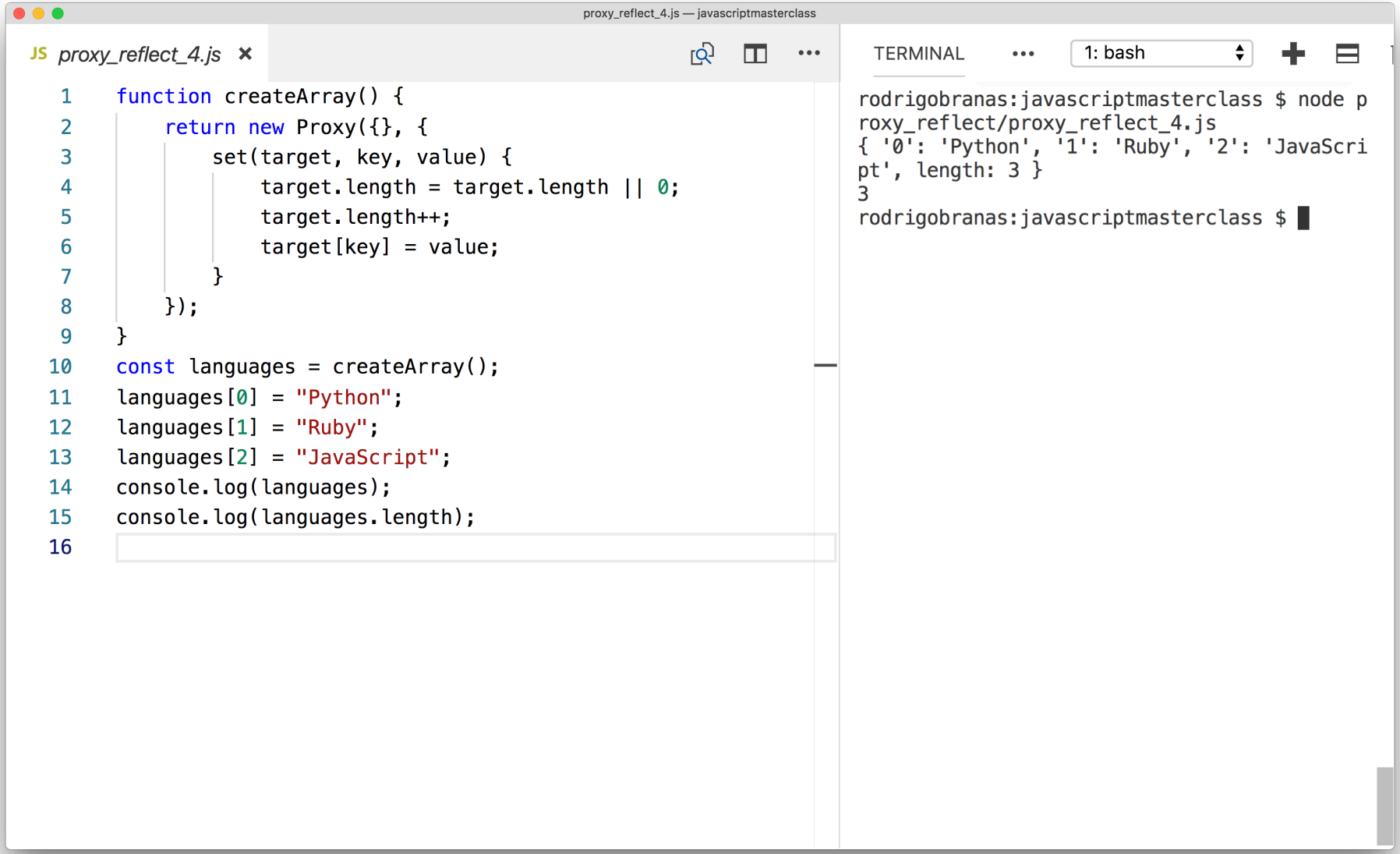

proxy_reflect_2.js — javascriptmasterclass
JS proxy_reflect_2.js
function createArray() {
return new Proxy({}, {
});
}
const languages = createArray();
languages[0] = "Python";
languages[1] = "Ruby";
languages[2] = "JavaScript";
console.log(languages);
console.log(languages.length);
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
roxy_reflect/proxy_reflect_2.js
{ '0': 'Python', '1': 'Ruby', '2': 'JavaScri
pt' }
undefined
rodrigobranas:javascriptmasterclass $

Existem métodos, chamados de trap, para diversos tipos de eventos relacionados a um objeto como:

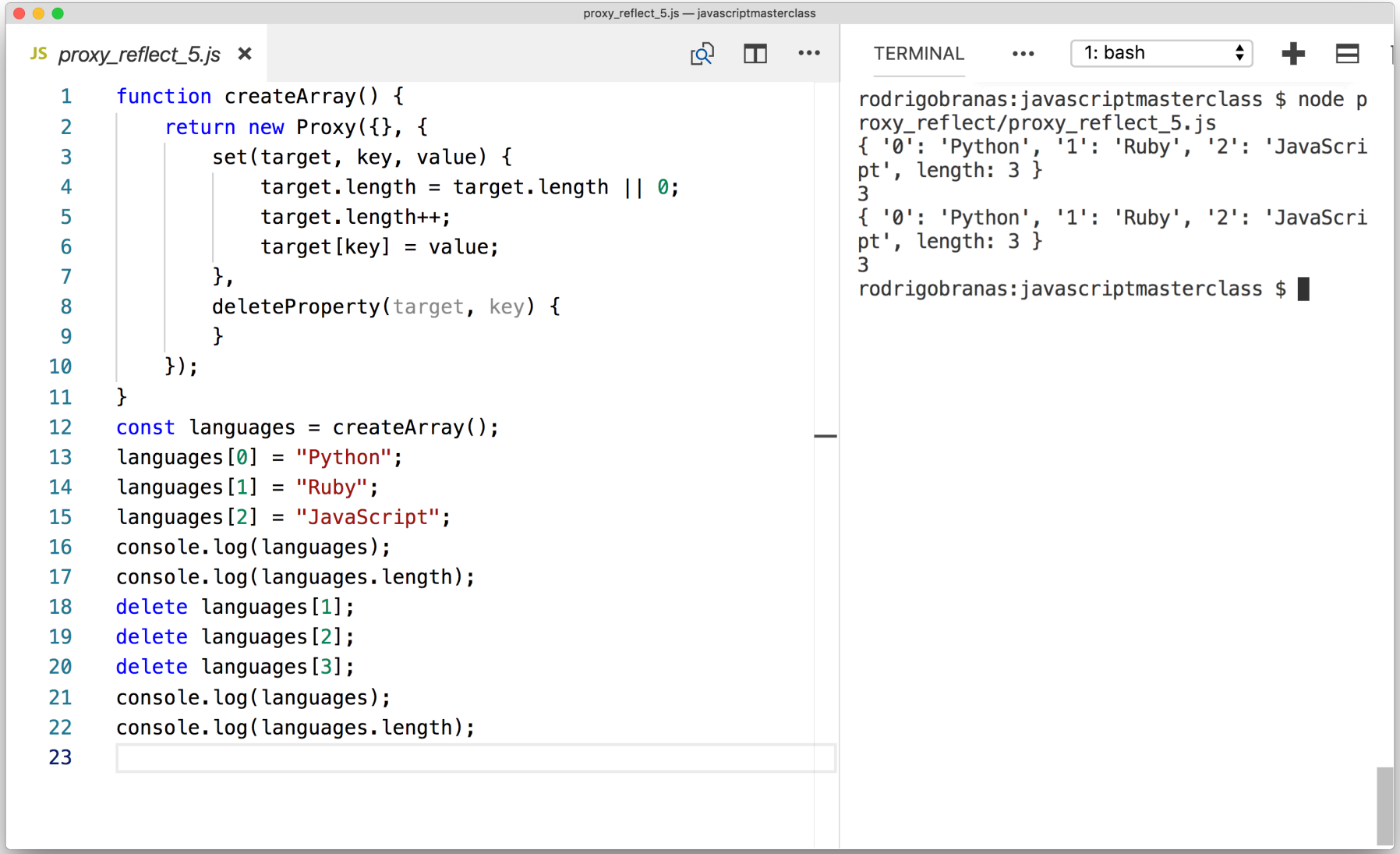
- apply
- construct
- defineProperty
- deleteProperty
- get
- getOwnPropertyDescriptor
- getPrototypeOf
- has
- isExtensible
- ownKeys
- preventExtensions
- set
- setPrototypeOf

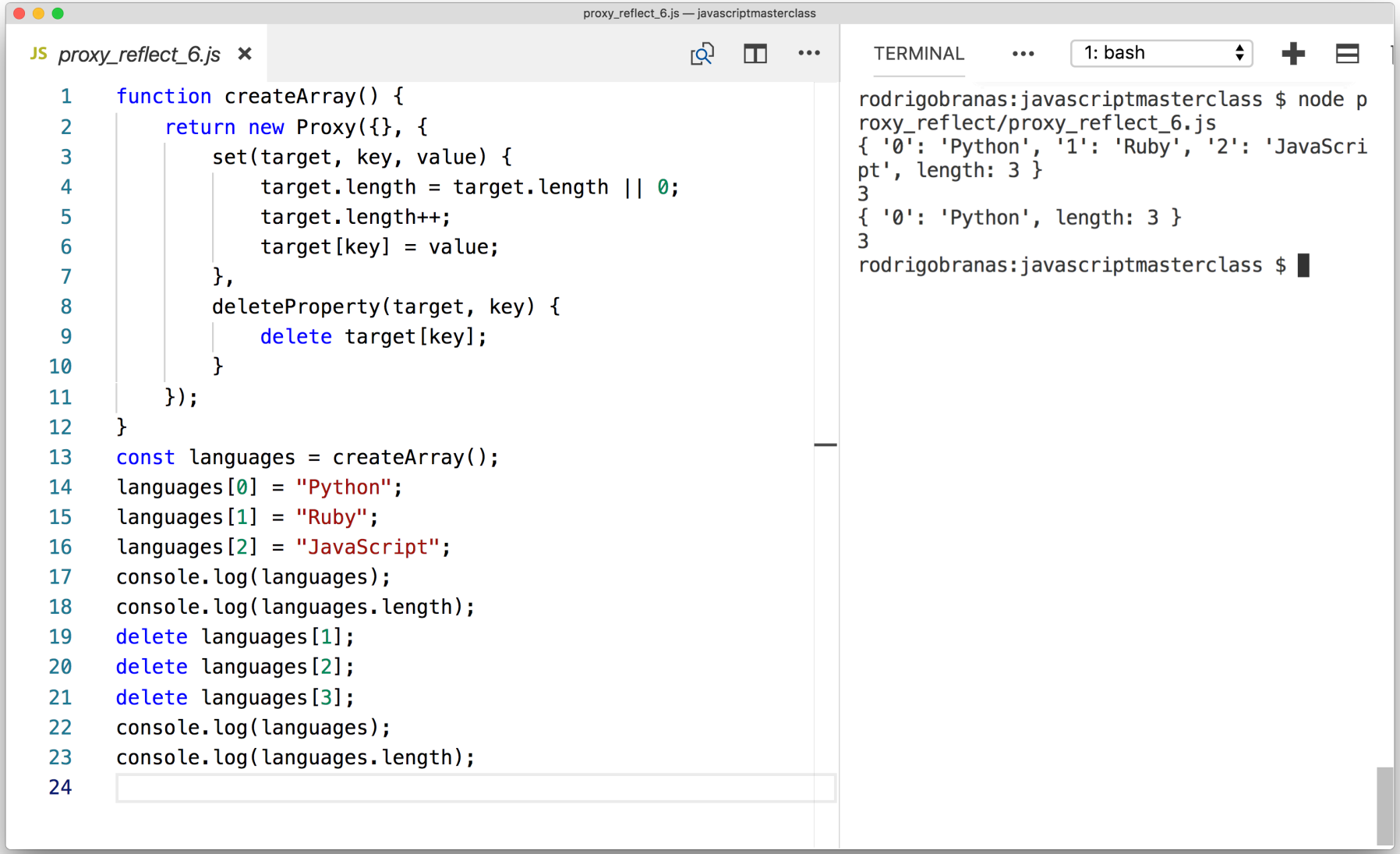
O método set é invocado quando uma propriedade é definida no objeto

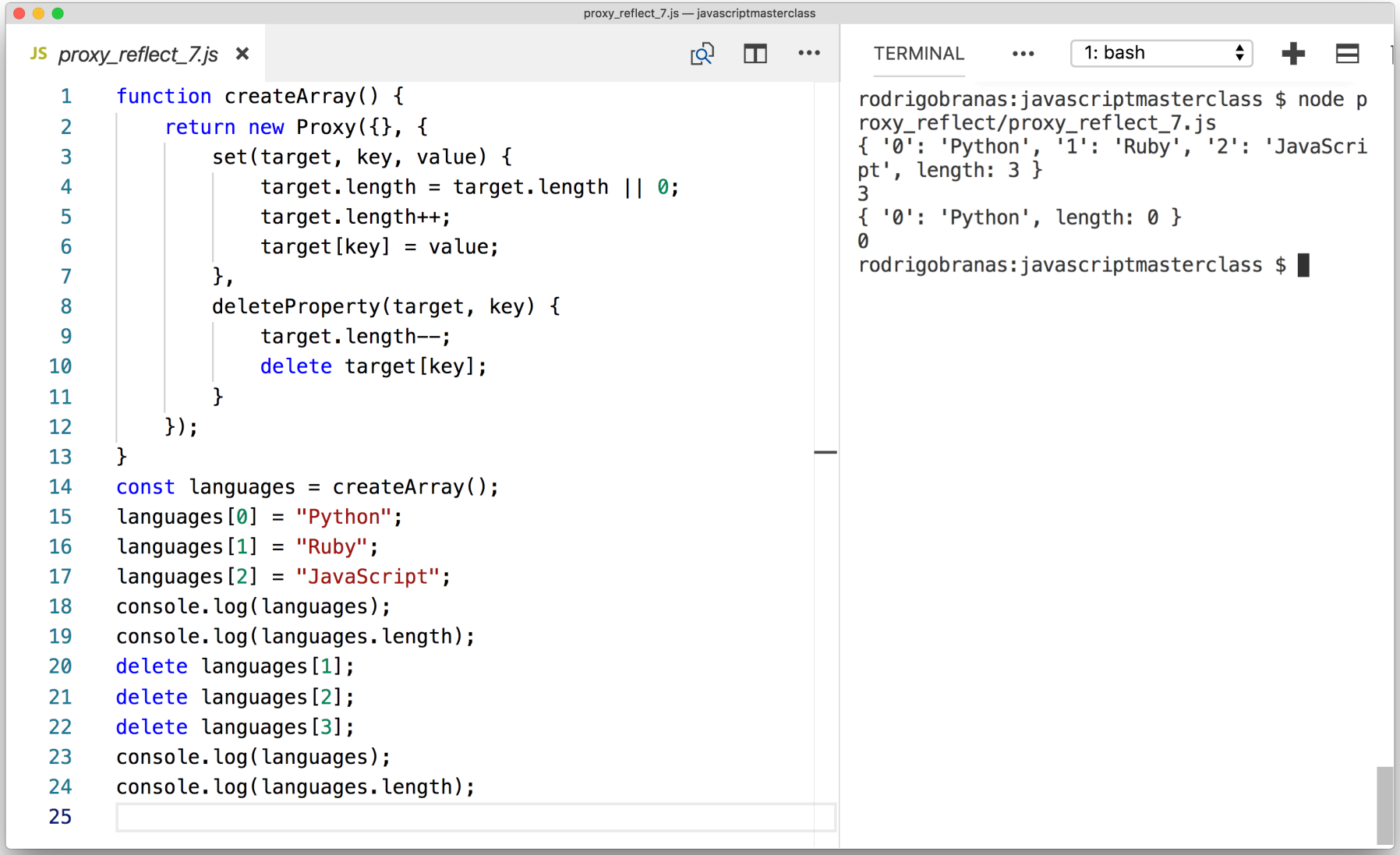

proxy_reflect_3.js — javascriptmasterclass
JS proxy_reflect_3.js
function createArray() {
return new Proxy({}, {
set(target, key, value) {
}
});
}
const languages = createArray();
languages[0] = "Python";
languages[1] = "Ruby";
languages[2] = "JavaScript";
console.log(languages);
console.log(languages.length);
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
roxy_reflect/proxy_reflect_3.js
{}
undefined
rodrigobranas:javascriptmasterclass $

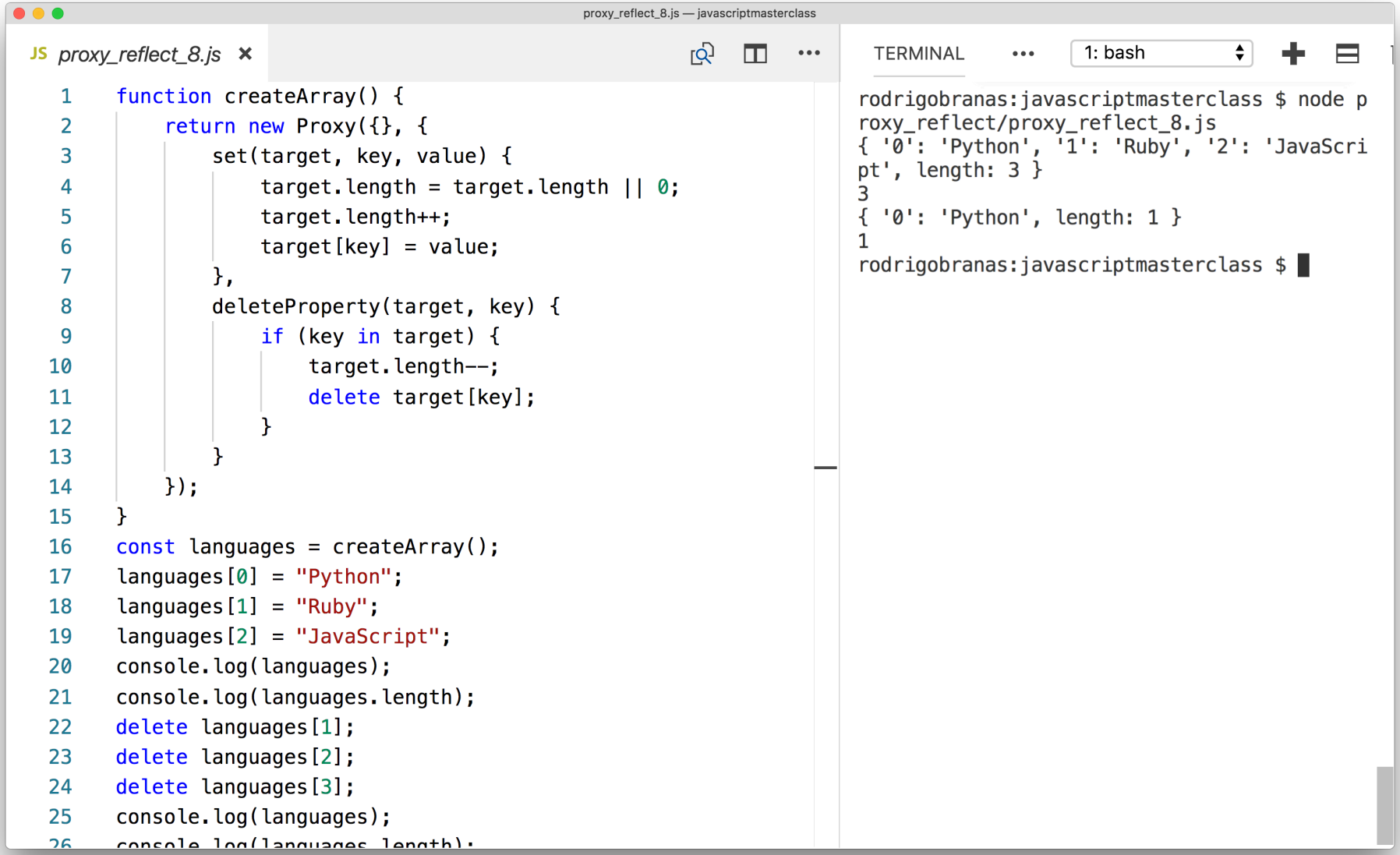

proxy_reflect_4.js — javascriptmasterclass
JS proxy_reflect_4.js
function createArray() {
return new Proxy({}, {
set(target, key, value) {
target.length = target.length || 0;
target.length++;
target[key] = value;
}
});
}
const languages = createArray();
languages[0] = "Python";
languages[1] = "Ruby";
languages[2] = "JavaScript";
console.log(languages);
console.log(languages.length);
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node proxy_reflect/proxy_reflect_4.js
{ '0': 'Python', '1': 'Ruby', '2': 'JavaScript', length: 3 }
3
rodrigobranas:javascriptmasterclass $

O método deleteProperty é invocado quando uma propriedade é deletada

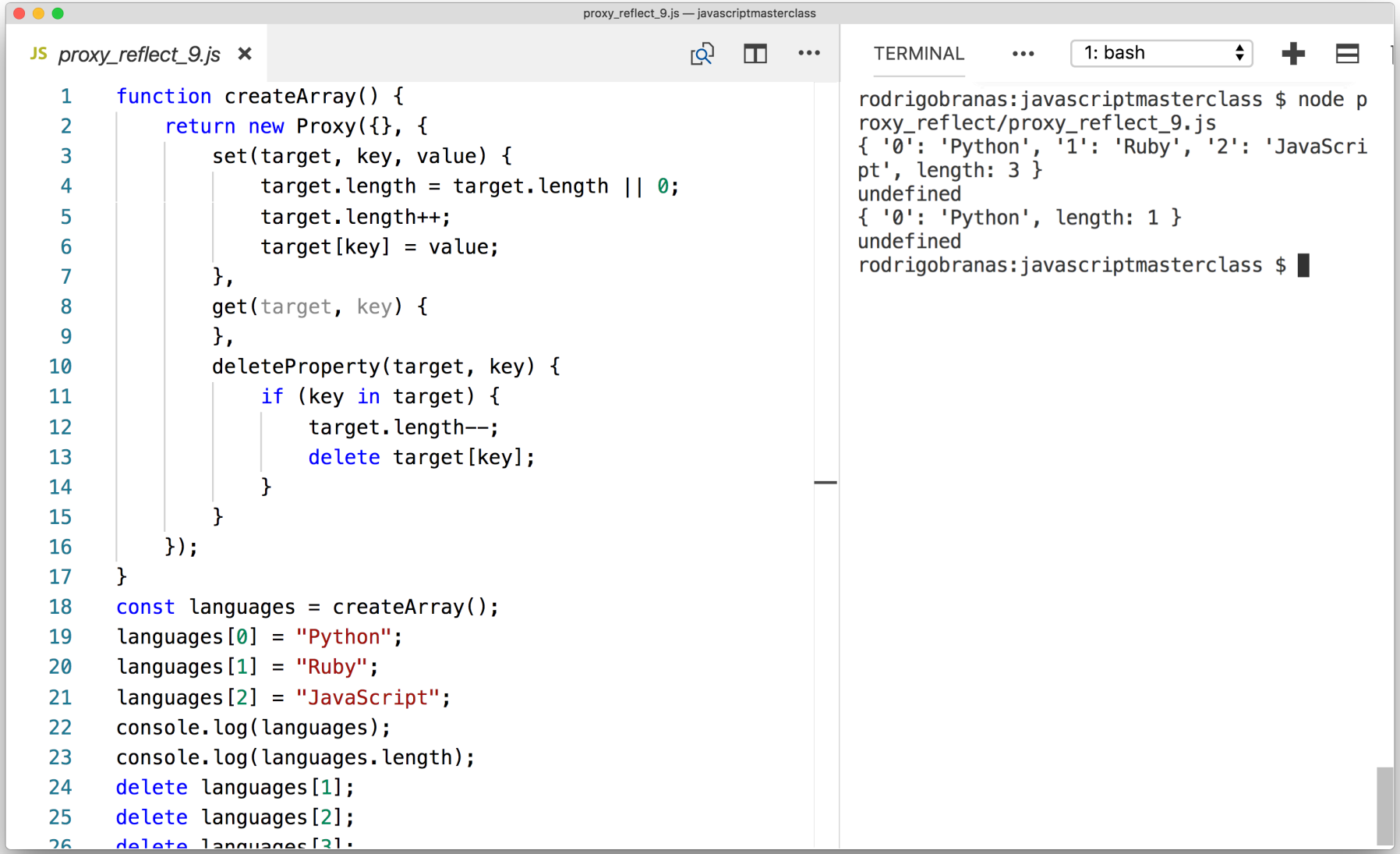

proxy_reflect_5.js — javascriptmasterclass
JS proxy_reflect_5.js
function createArray() {
return new Proxy({}, {
set(target, key, value) {
target.length = target.length || 0;
target.length++;
target[key] = value;
},
deleteProperty(target, key) {
}
});
}
const languages = createArray();
languages[0] = "Python";
languages[1] = "Ruby";
languages[2] = "JavaScript";
console.log(languages);
console.log(languages.length);
delete languages[1];
delete languages[2];
delete languages[3];
console.log(languages);
console.log(languages.length);
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
roxy_reflect/proxy_reflect_5.js
{ '0': 'Python', '1': 'Ruby', '2': 'JavaScri
pt', length: 3 }
3
{ '0': 'Python', '1': 'Ruby', '2': 'JavaScri
pt', length: 3 }
3
rodrigobranas:javascriptmasterclass $

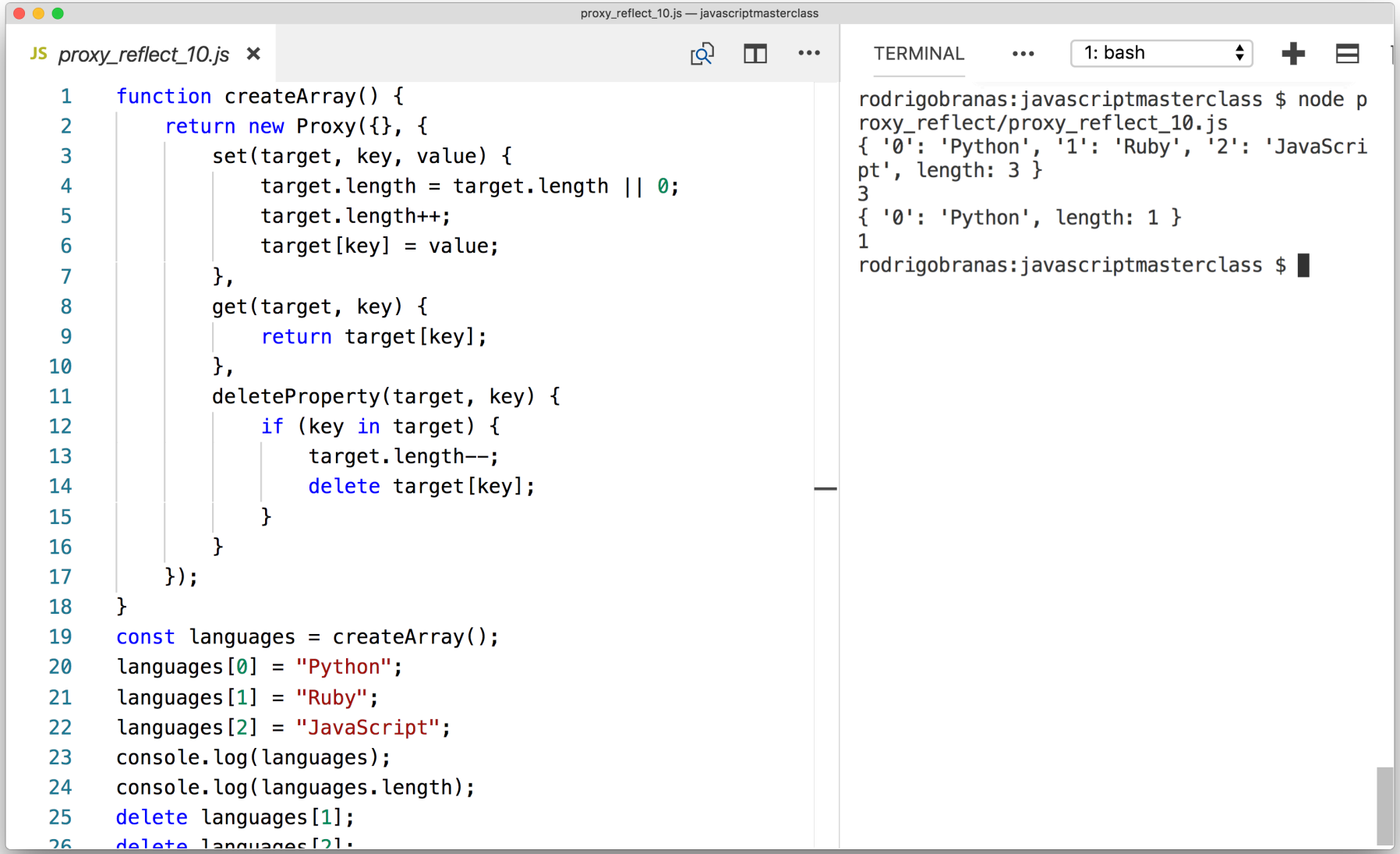

proxy_reflect_6.js — javascriptmasterclass
proxy_reflect_6.js
function createArray() {
return new Proxy({}, {
set(target, key, value) {
target.length = target.length || 0;
target.length++;
target[key] = value;
},
deleteProperty(target, key) {
delete target[key];
}
});
}
const languages = createArray();
languages[0] = "Python";
languages[1] = "Ruby";
languages[2] = "JavaScript";
console.log(languages);
console.log(languages.length);
delete languages[1];
delete languages[2];
delete languages[3];
console.log(languages);
console.log(languages.length);
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node proxy_reflect/proxy_reflect_6.js
{ '0': 'Python', '1': 'Ruby', '2': 'JavaScript', length: 3 }
3
{ '0': 'Python', length: 3 }
3
rodrigobranas:javascriptmasterclass $

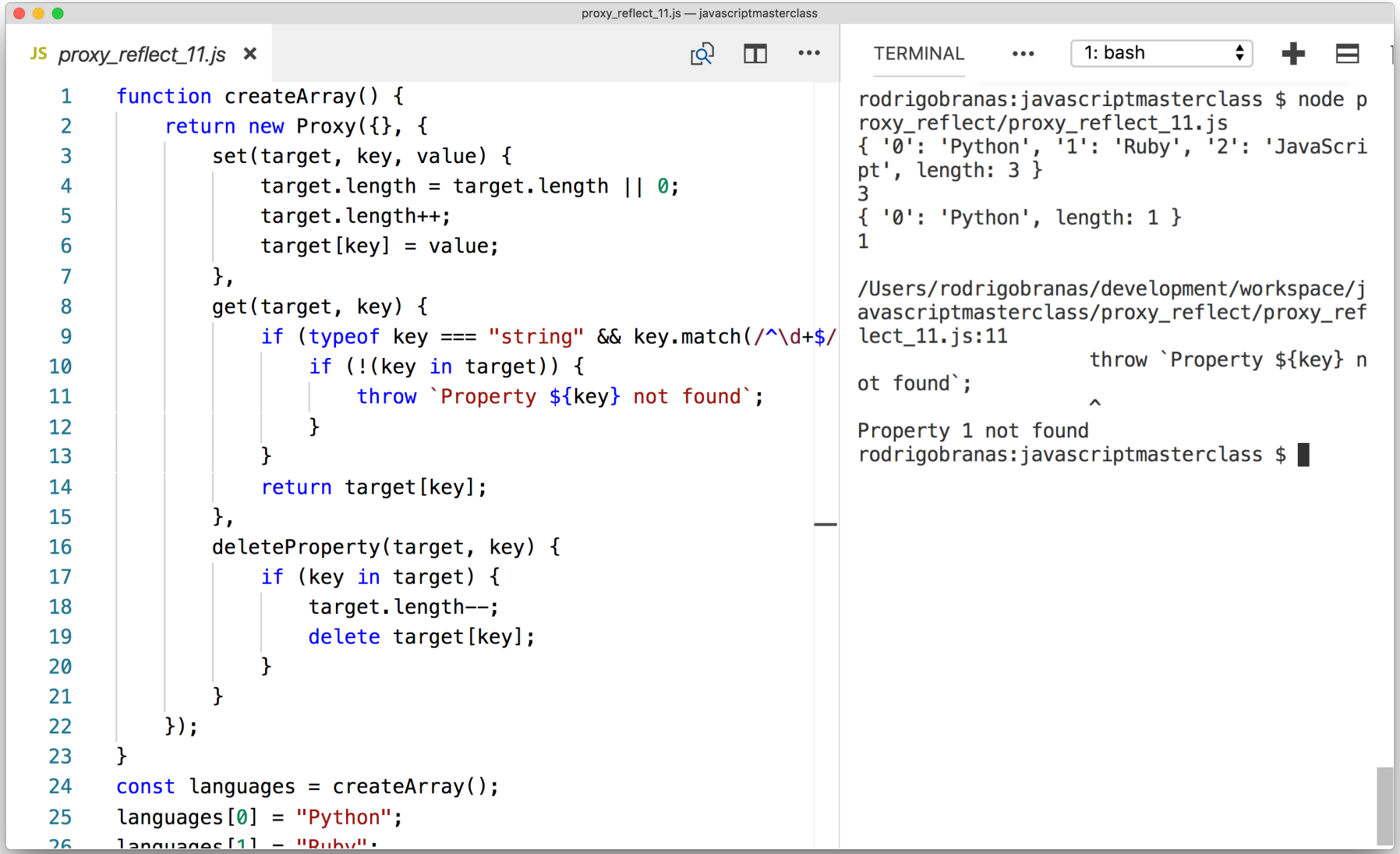

proxy_reflect_7.js — javascriptmasterclass
JS proxy_reflect_7.js
function createArray() {
return new Proxy({}, {
set(target, key, value) {
target.length = target.length || 0;
target.length++;
target[key] = value;
},
deleteProperty(target, key) {
target.length--;
delete target[key];
}
});
}
const languages = createArray();
languages[0] = "Python";
languages[1] = "Ruby";
languages[2] = "JavaScript";
console.log(languages);
console.log(languages.length);
delete languages[1];
delete languages[2];
delete languages[3];
console.log(languages);
console.log(languages.length);
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
roxy_reflect/proxy_reflect_7.js
{ '0': 'Python', '1': 'Ruby', '2': 'JavaScri
pt', length: 3 }
3
{ '0': 'Python', length: 0 }
0
rodrigobranas:javascriptmasterclass $


proxy_reflect_8.js — javascriptmasterclass
proxy_reflect_8.js
function createArray() {
return new Proxy({}, {
set(target, key, value) {
target.length = target.length || 0;
target.length++;
target[key] = value;
},
deleteProperty(target, key) {
if (key in target) {
target.length--;
delete target[key];
}
}
});
}
const languages = createArray();
languages[0] = "Python";
languages[1] = "Ruby";
languages[2] = "JavaScript";
console.log(languages);
console.log(languages.length);
delete languages[1];
delete languages[2];
delete languages[3];
console.log(languages);
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
roxy_reflect/proxy_reflect_8.js
{ '0': 'Python', '1': 'Ruby', '2': 'JavaScri
pt', length: 3 }
3
{ '0': 'Python', length: 1 }
1
rodrigobranas:javascriptmasterclass $

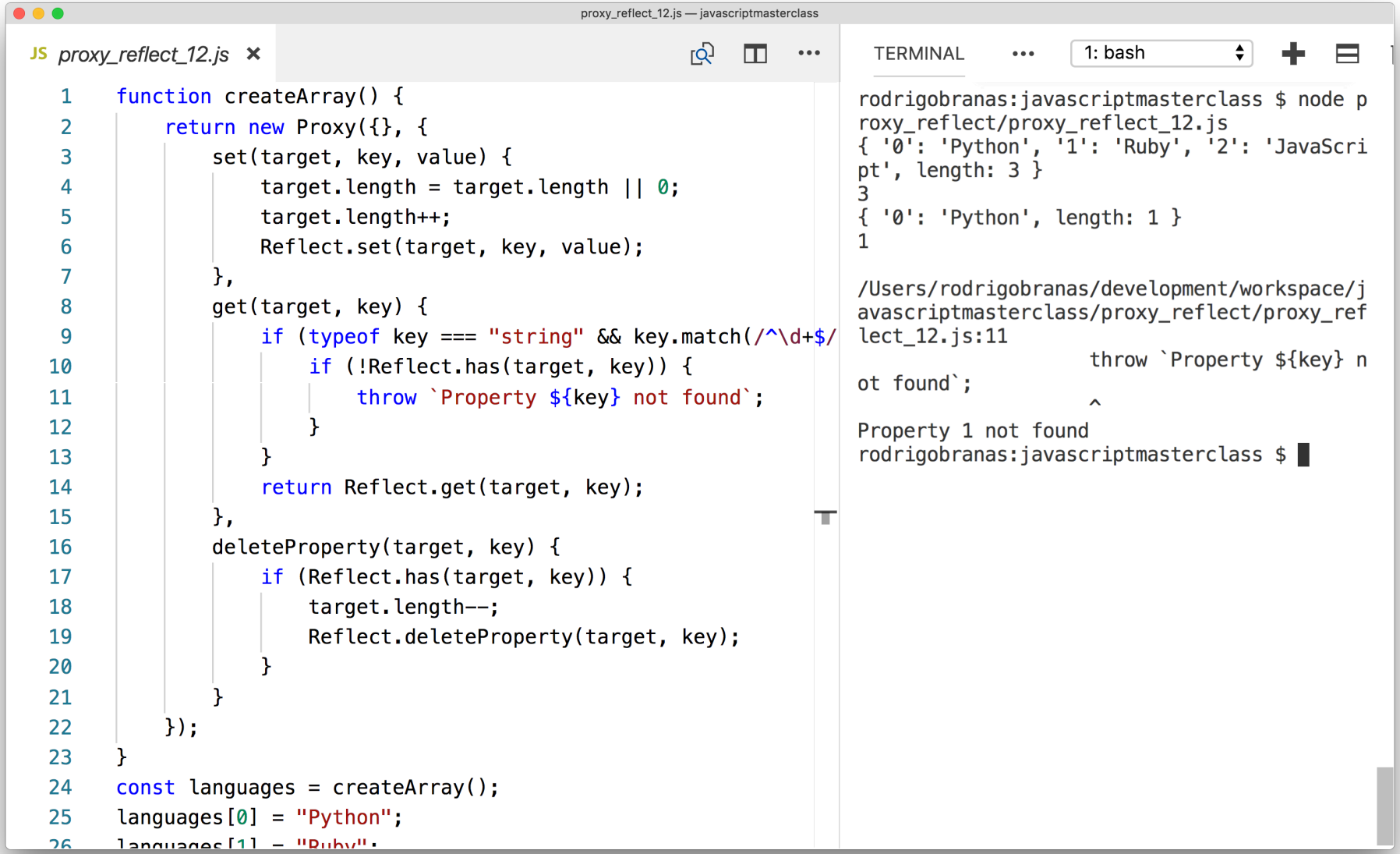
O método get é invocado quando uma propriedade é acessada


proxy_reflect_9.js — javascriptmasterclass
proxy_reflect_9.js
function createArray() {
return new Proxy({}, {
set(target, key, value) {
target.length = target.length || 0;
target.length++;
target[key] = value;
},
get(target, key) {
},
deleteProperty(target, key) {
if (key in target) {
target.length--;
delete target[key];
}
}
});
}
const languages = createArray();
languages[0] = "Python";
languages[1] = "Ruby";
languages[2] = "JavaScript";
console.log(languages);
console.log(languages.length);
delete languages[1];
delete languages[2];
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
roxy_reflect/proxy_reflect_9.js
{ '0': 'Python', '1': 'Ruby', '2': 'JavaScri
pt', length: 3 }
undefined
{ '0': 'Python', length: 1 }
undefined
rodrigobranas:javascriptmasterclass $


proxy_reflect_10.js — javascriptmasterclass
proxy_reflect_10.js
function createArray() {
return new Proxy({}, {
set(target, key, value) {
target.length = target.length || 0;
target.length++;
target[key] = value;
},
get(target, key) {
return target[key];
},
deleteProperty(target, key) {
if (key in target) {
target.length--;
delete target[key];
}
}
});
}
const languages = createArray();
languages[0] = "Python";
languages[1] = "Ruby";
languages[2] = "JavaScript";
console.log(languages);
console.log(languages.length);
delete languages[1];
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
roxy_reflect/proxy_reflect_10.js
{ '0': 'Python', '1': 'Ruby', '2': 'JavaScri
pt', length: 3 }
3
{ '0': 'Python', length: 1 }
1
rodrigobranas:javascriptmasterclass $


proxy_reflect_11.js — javascriptmasterclass
proxy_reflect_11.js
function createArray() {
return new Proxy({}, {
set(target, key, value) {
target.length = target.length || 0;
target.length++;
target[key] = value;
},
get(target, key) {
if (typeof key === "string" && key.match(/^\d+$/
if (!(key in target)) {
throw `Property ${key} not found`;
}
}
return target[key];
},
deleteProperty(target, key) {
if (key in target) {
target.length--;
delete target[key];
}
}
});
}
const languages = createArray();
languages[0] = "Python";
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
roxy_reflect/proxy_reflect_11.js
{ '0': 'Python', '1': 'Ruby', '2': 'JavaScri
pt', length: 3 }
3
{ '0': 'Python', length: 1 }
1
/Users/rodrigobranas/development/workspace/j
avascriptmasterclass/proxy_reflect/proxy_ref
lect_11.js:11
throw `Property ${key} n
ot found`;
^
Property 1 not found
rodrigobranas:javascriptmasterclass $

A Reflect API tem os mesmos métodos que existem no Proxy, permitindo a execução de diversos tipos de operações no objeto alvo


proxy_reflect_12.js — javascriptmasterclass
proxy_reflect_12.js
function createArray() {
return new Proxy({}, {
set(target, key, value) {
target.length = target.length || 0;
target.length++;
Reflect.set(target, key, value);
},
get(target, key) {
if (typeof key === "string" && key.match(/^\d+$/
if (!Reflect.has(target, key)) {
throw `Property ${key} not found`;
}
}
return Reflect.get(target, key);
},
deleteProperty(target, key) {
if (Reflect.has(target, key)) {
target.length--;
Reflect.deleteProperty(target, key);
}
}
});
}
const languages = createArray();
languages[0] = "Python";
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
roxy_reflect/proxy_reflect_12.js
{ '0': 'Python', '1': 'Ruby', '2': 'JavaScri
pt', length: 3 }
3
{ '0': 'Python', length: 1 }
1
/Users/rodrigobranas/development/workspace/j
avascriptmasterclass/proxy_reflect/proxy_ref
lect_12.js:11
throw `Property ${key} n
ot found`;
^
Property 1 not found
rodrigobranas:javascriptmasterclass $