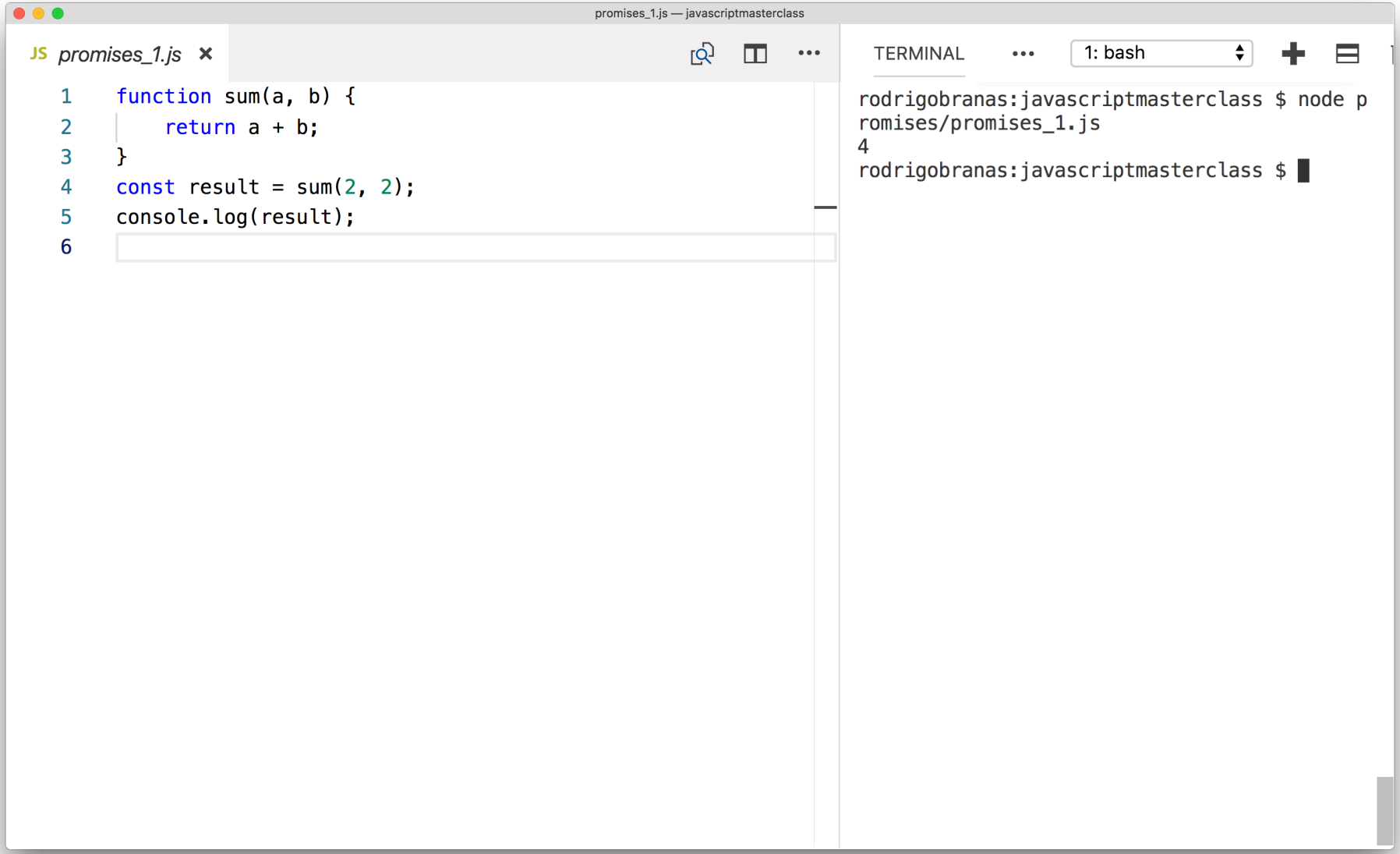
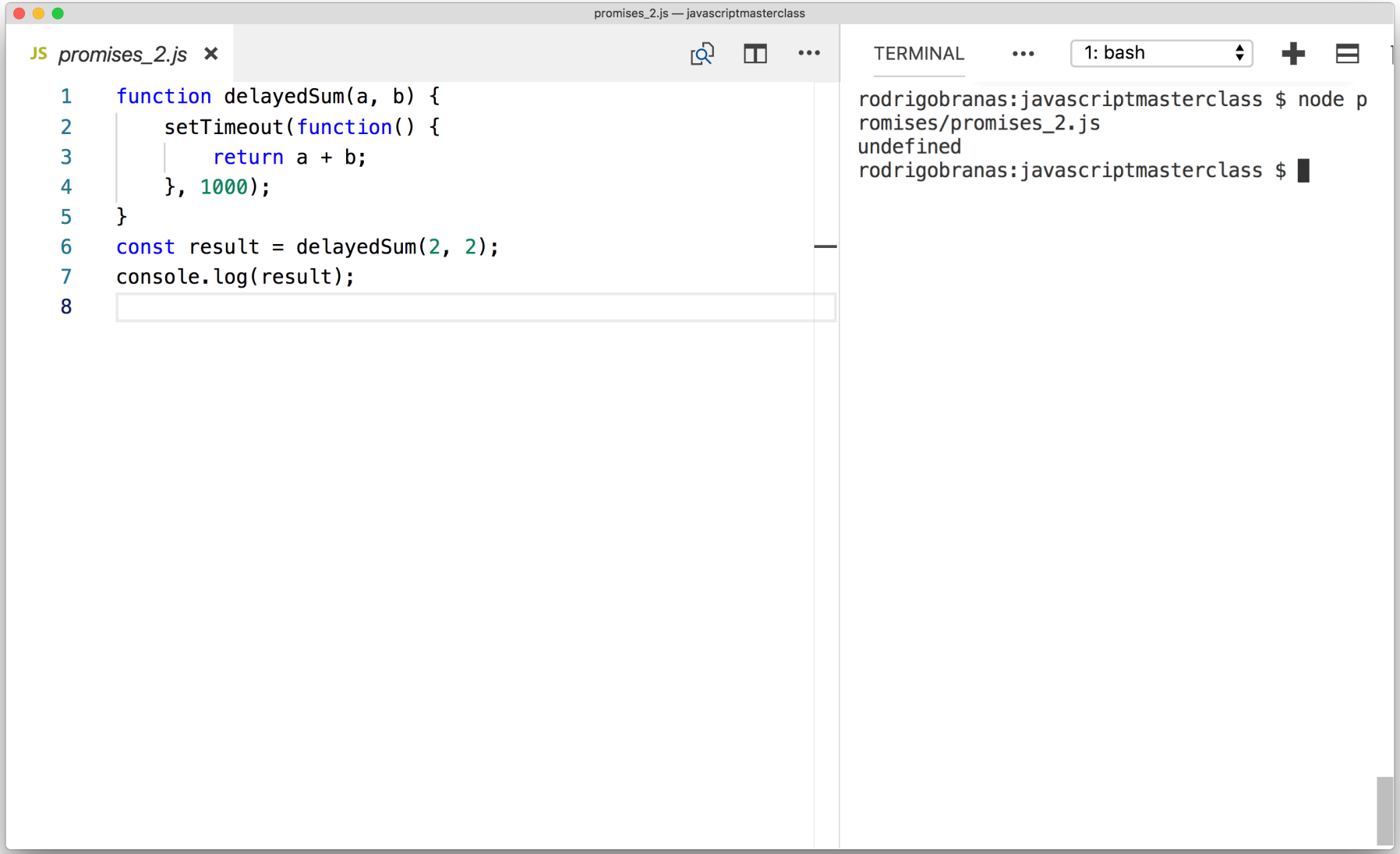
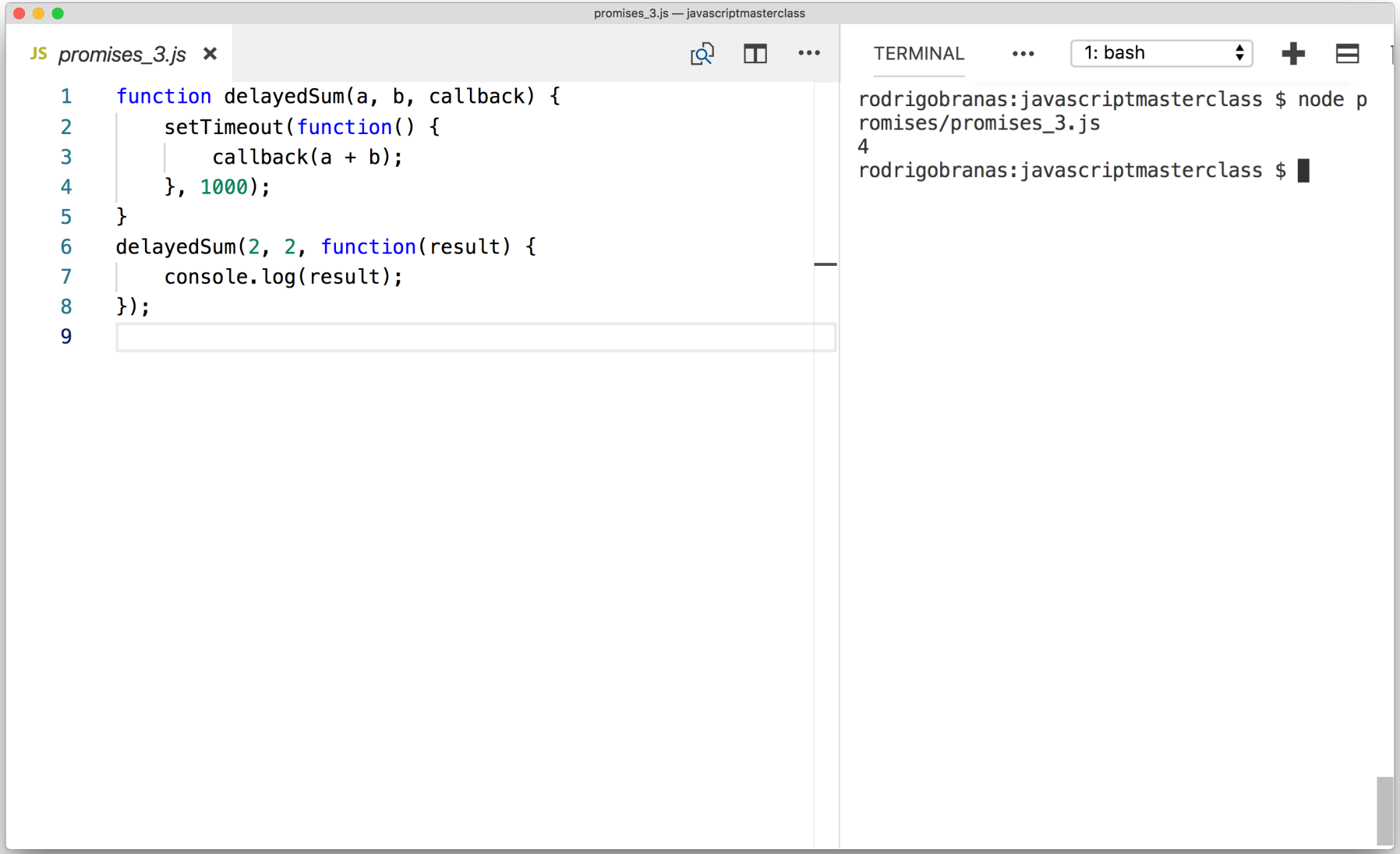
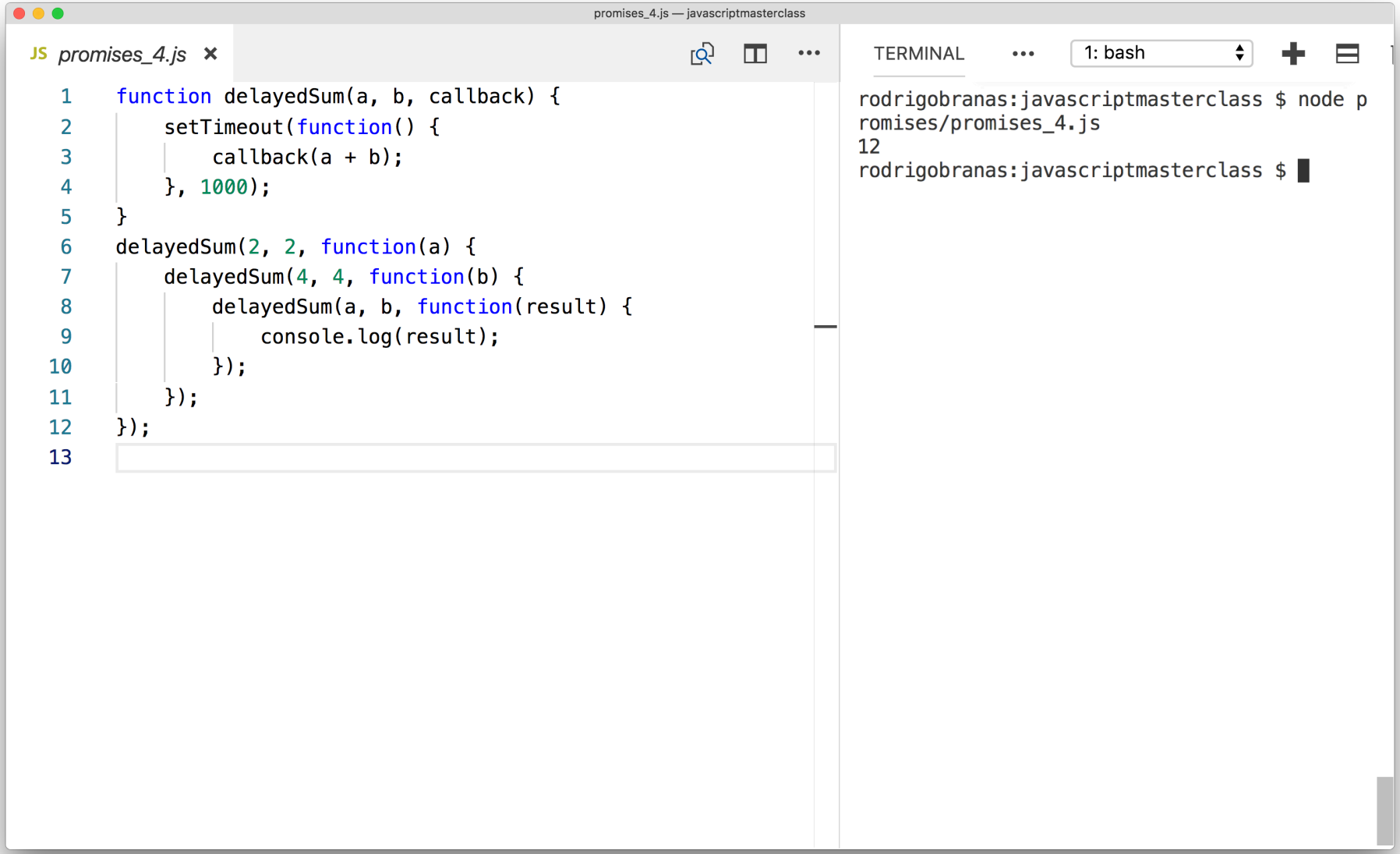

Promises

As promises são objetos responsáveis por modelar comportamento assíncrono, permitindo o seu tratamento de uma forma mais fácil e direta

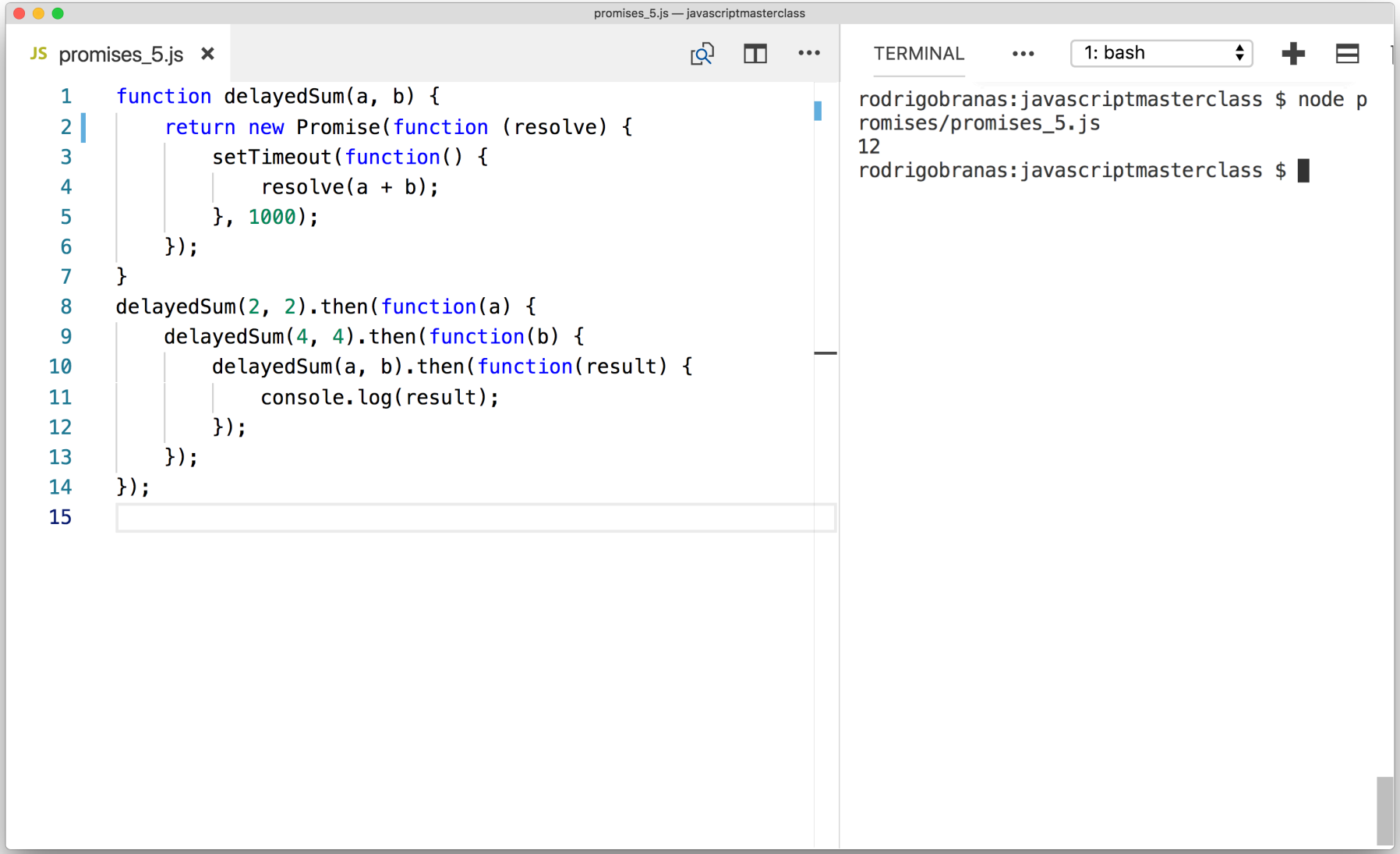

promises_1.js — javascriptmasterclass
JS promises_1.js
function sum(a, b) {
return a + b;
}
const result = sum(2, 2);
console.log(result);
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
romises/promises_1.js
4
rodrigobranas:javascriptmasterclass $


promises_2.js — javascriptmasterclass
promises_2.js
function delayedSum(a, b) {
setTimeout(function() {
return a + b;
}, 1000);
}
const result = delayedSum(2, 2);
console.log(result);
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
romises/promises_2.js
undefined
rodrigobranas:javascriptmasterclass $

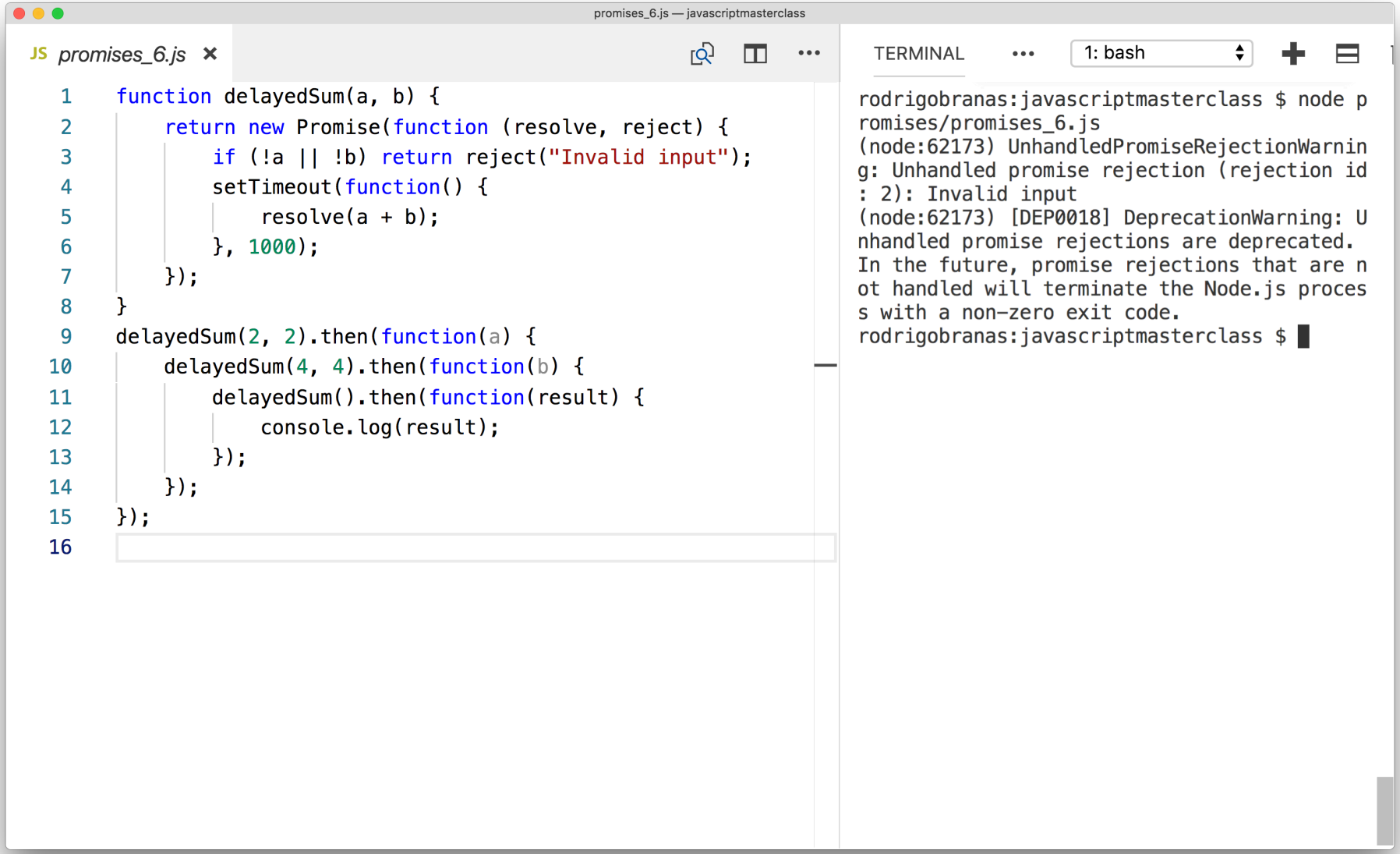

promises_3.js — javascriptmasterclass
promises_3.js
function delayedSum(a, b, callback) {
setTimeout(function() {
callback(a + b);
}, 1000);
}
delayedSum(2, 2, function(result) {
console.log(result);
});
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
romises/promises_3.js
4
rodrigobranas:javascriptmasterclass $

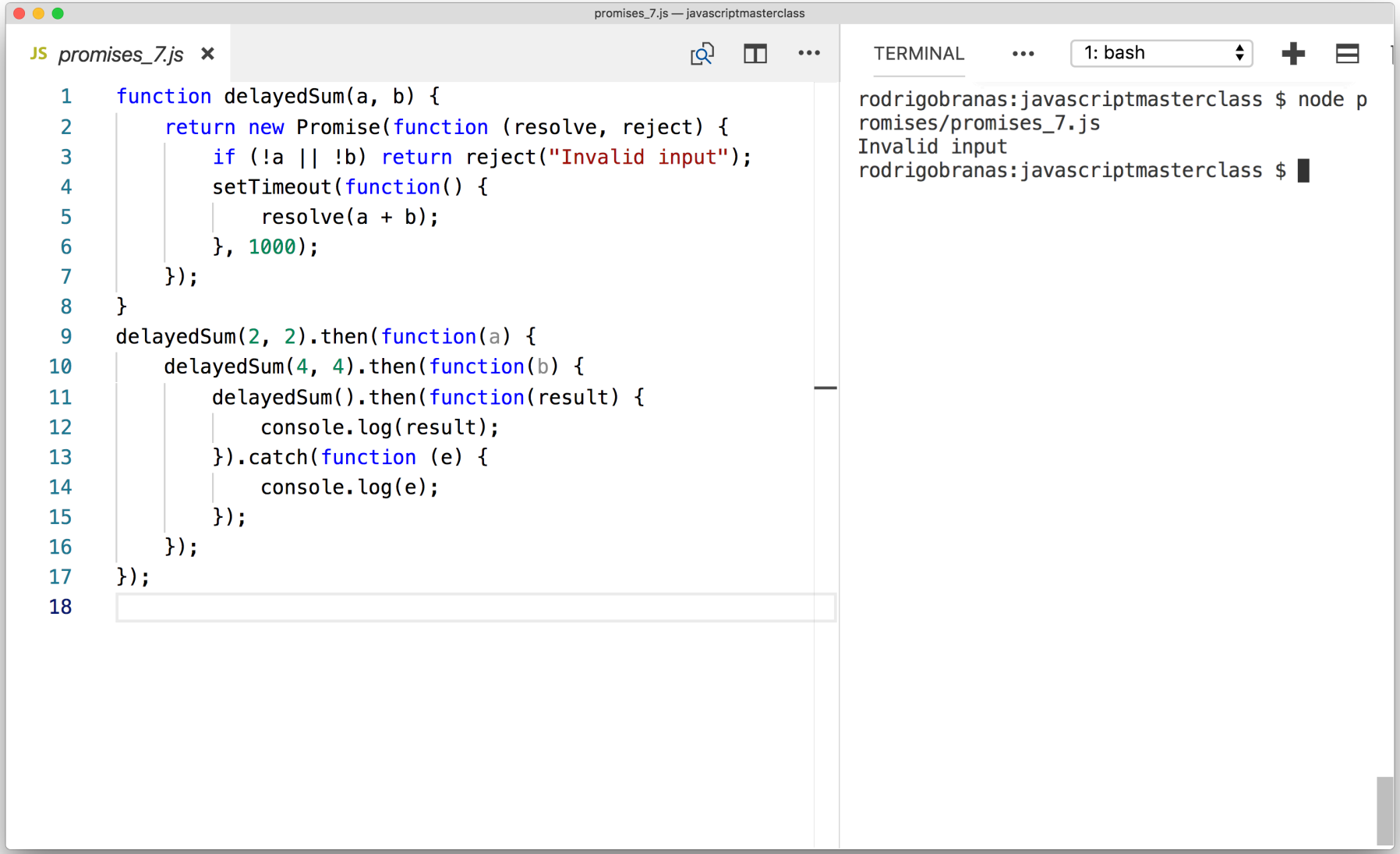

promises_4.js — javascriptmasterclass
promises_4.js
function delayedSum(a, b, callback) {
setTimeout(function() {
callback(a + b);
}, 1000);
}
delayedSum(2, 2, function(a) {
delayedSum(4, 4, function(b) {
delayedSum(a, b, function(result) {
console.log(result);
});
});
});
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
romises/promises_4.js
12
rodrigobranas:javascriptmasterclass $

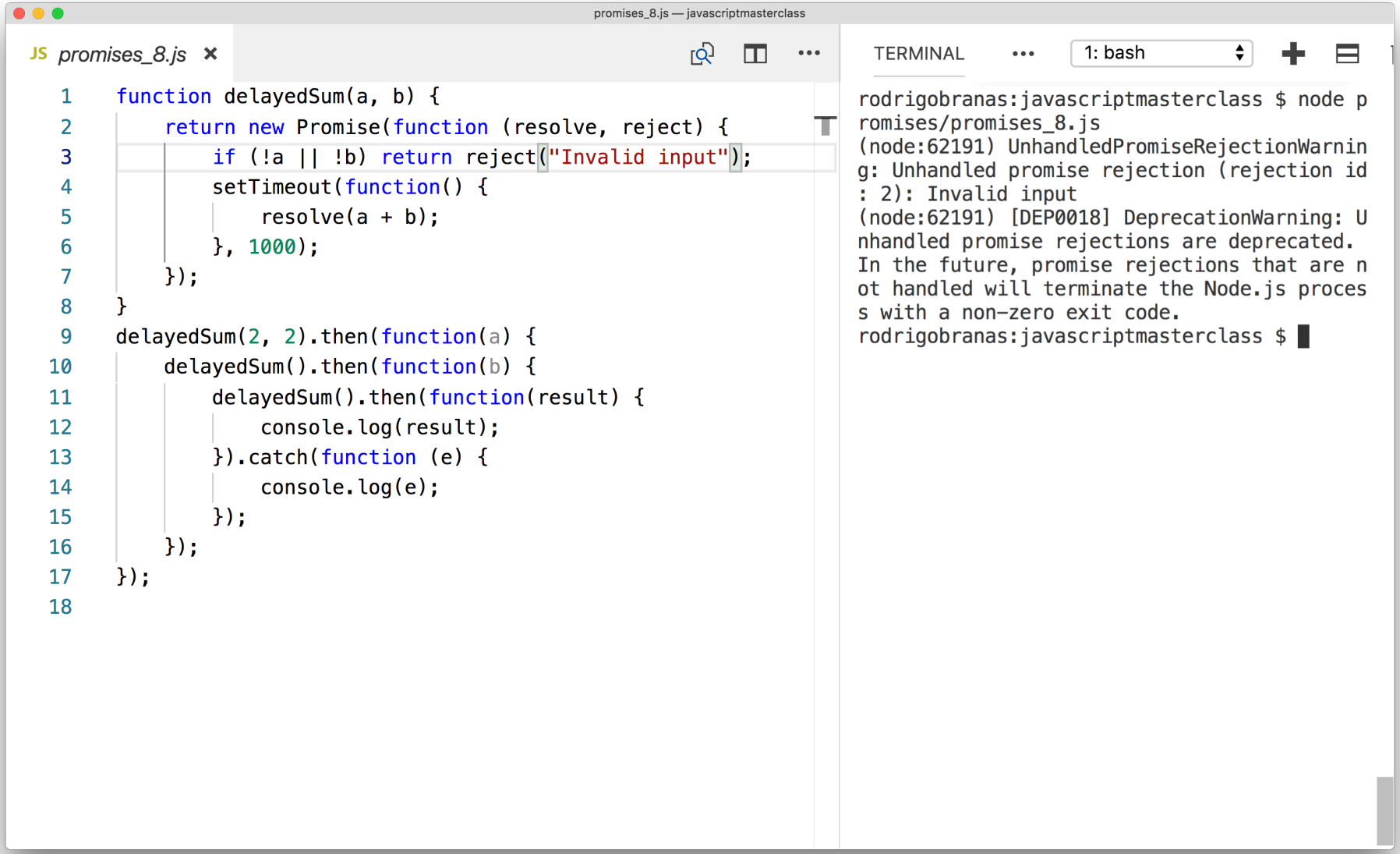
Para criar uma promise basta instanciá-la, executando a função resolve em caso de sucesso, sendo tratado por meio de then

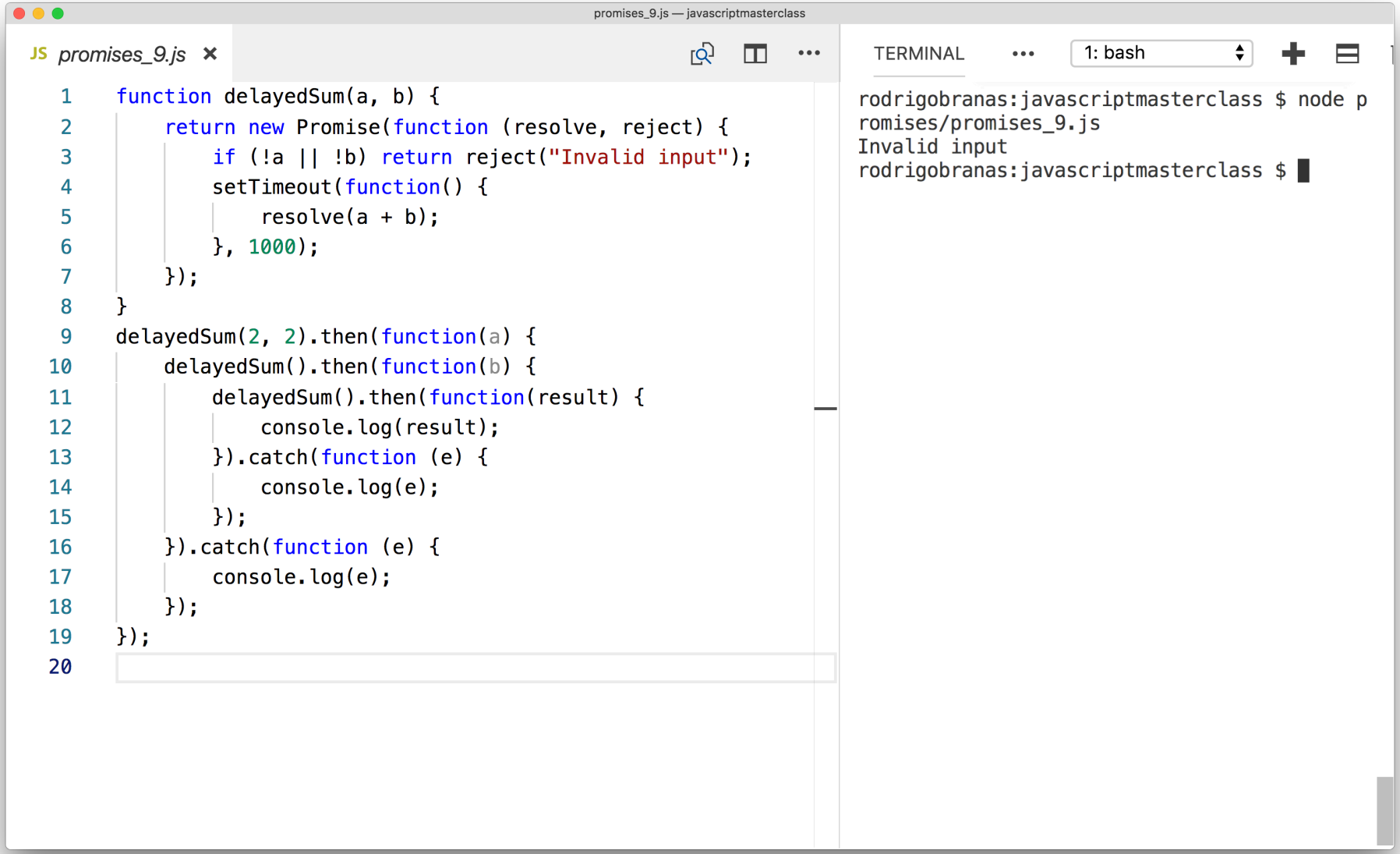

promises_5.js — javascriptmasterclass
promises_5.js
function delayedSum(a, b) {
return new Promise(function (resolve) {
setTimeout(function() {
resolve(a + b);
}, 1000);
});
}
delayedSum(2, 2).then(function(a) {
delayedSum(4, 4).then(function(b) {
delayedSum(a, b).then(function(result) {
console.log(result);
});
});
});
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
romises/promises_5.js
12
rodrigobranas:javascriptmasterclass $

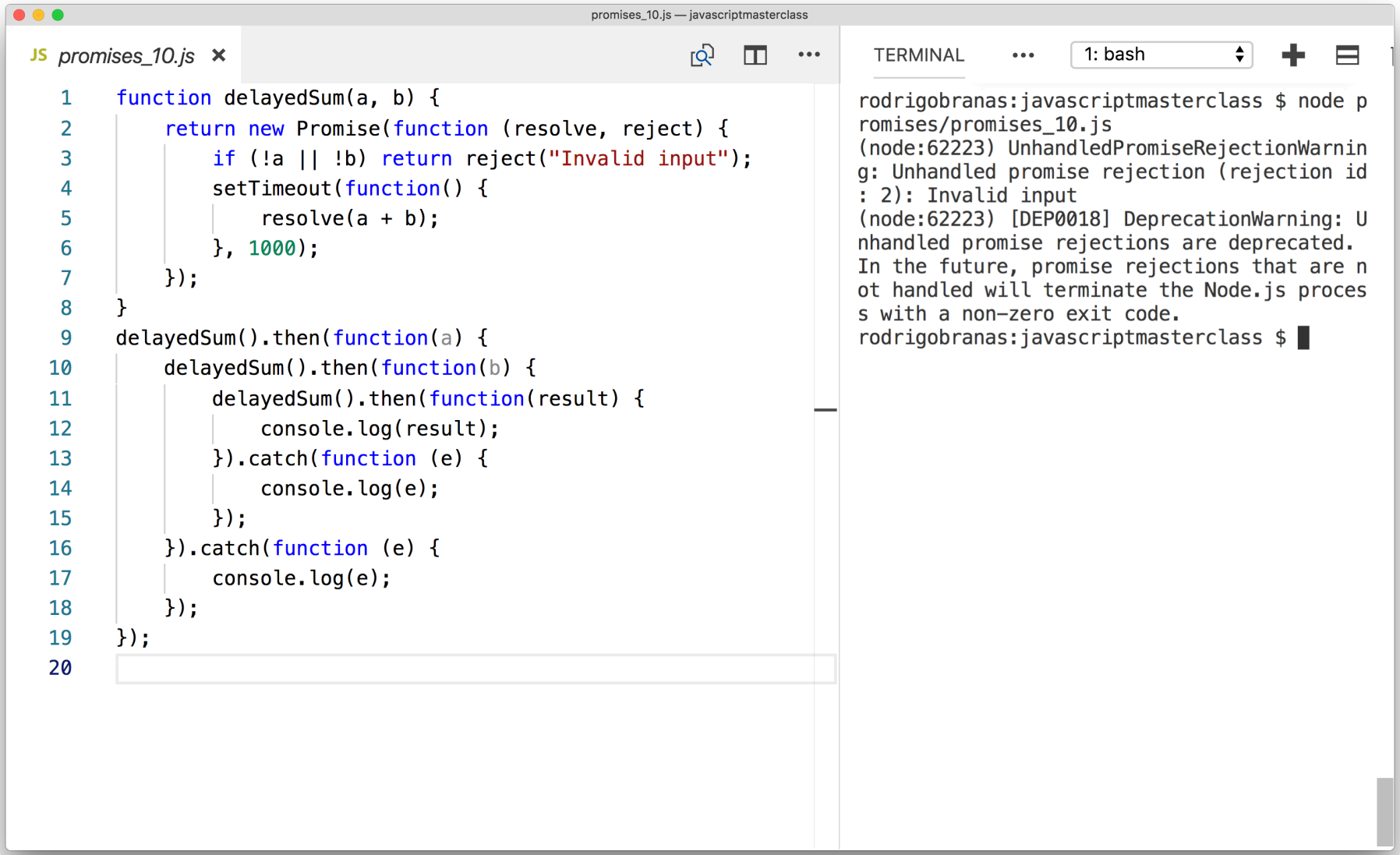
Em caso de fracasso, a função reject deve ser executada, sendo tratada por meio de catch

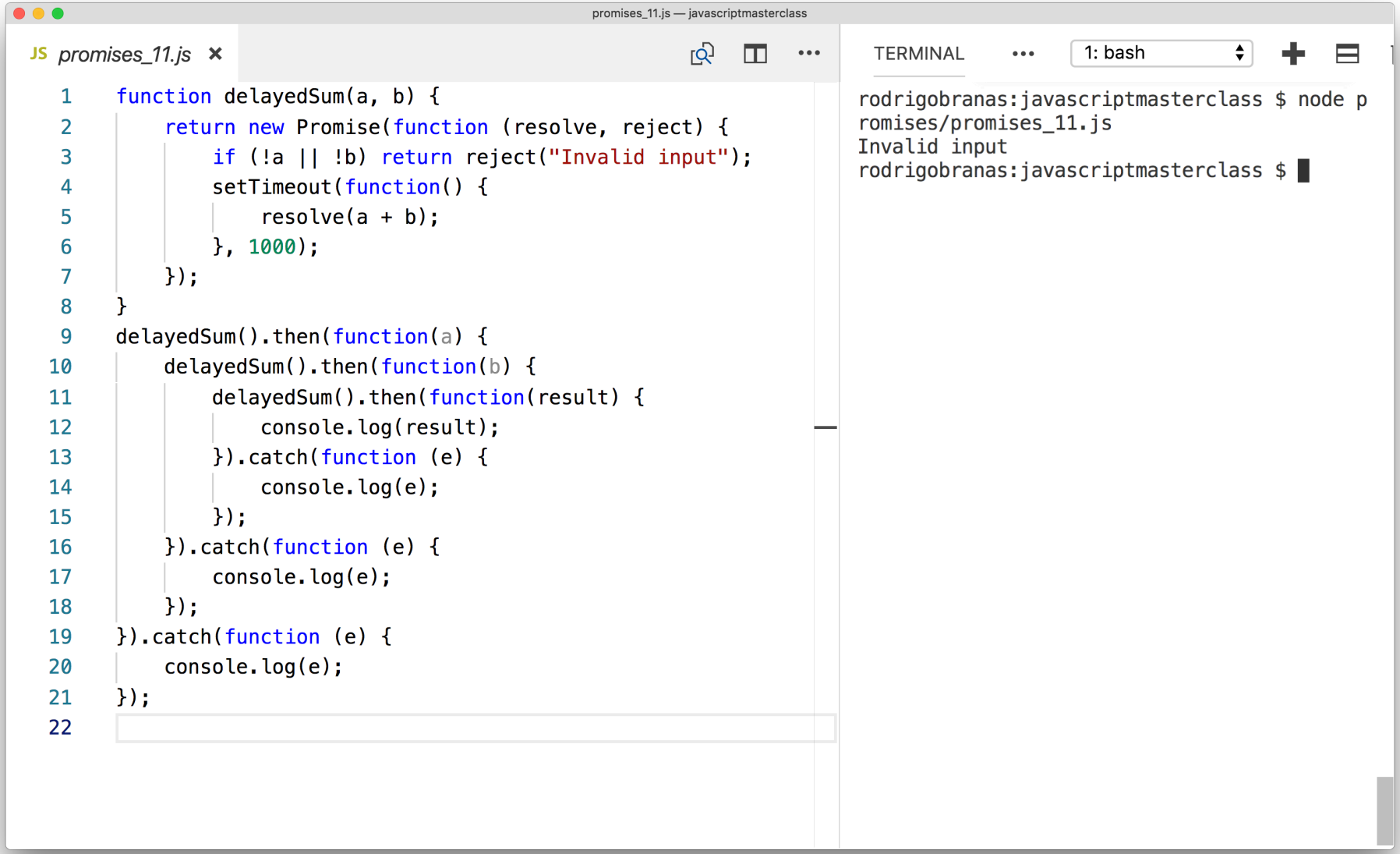

promises_6.js — javascriptmasterclass
promises_6.js
function delayedSum(a, b) {
return new Promise(function (resolve, reject) {
if (!a || !b) return reject("Invalid input");
setTimeout(function() {
resolve(a + b);
}, 1000);
});
}
delayedSum(2, 2).then(function(a) {
delayedSum(4, 4).then(function(b) {
delayedSum().then(function(result) {
console.log(result);
});
});
});
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node promises/promises_6.js
(node:62173) UnhandledPromiseRejectionWarning: Unhandled promise rejection (rejection id: 2): Invalid input
(node:62173) [DEP0018] DeprecationWarning: Unhandled promise rejections are deprecated. In the future, promise rejections that are not handled will terminate the Node.js process with a non-zero exit code.
rodrigobranas:javascriptmasterclass $


promises_7.js — javascriptmasterclass
promises_7.js
function delayedSum(a, b) {
return new Promise(function (resolve, reject) {
if (!a || !b) return reject("Invalid input");
setTimeout(function() {
resolve(a + b);
}, 1000);
});
}
delayedSum(2, 2).then(function(a) {
delayedSum(4, 4).then(function(b) {
delayedSum().then(function(result) {
console.log(result);
}).catch(function (e) {
console.log(e);
});
});
});
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
romises/promises_7.js
Invalid input
rodrigobranas:javascriptmasterclass $

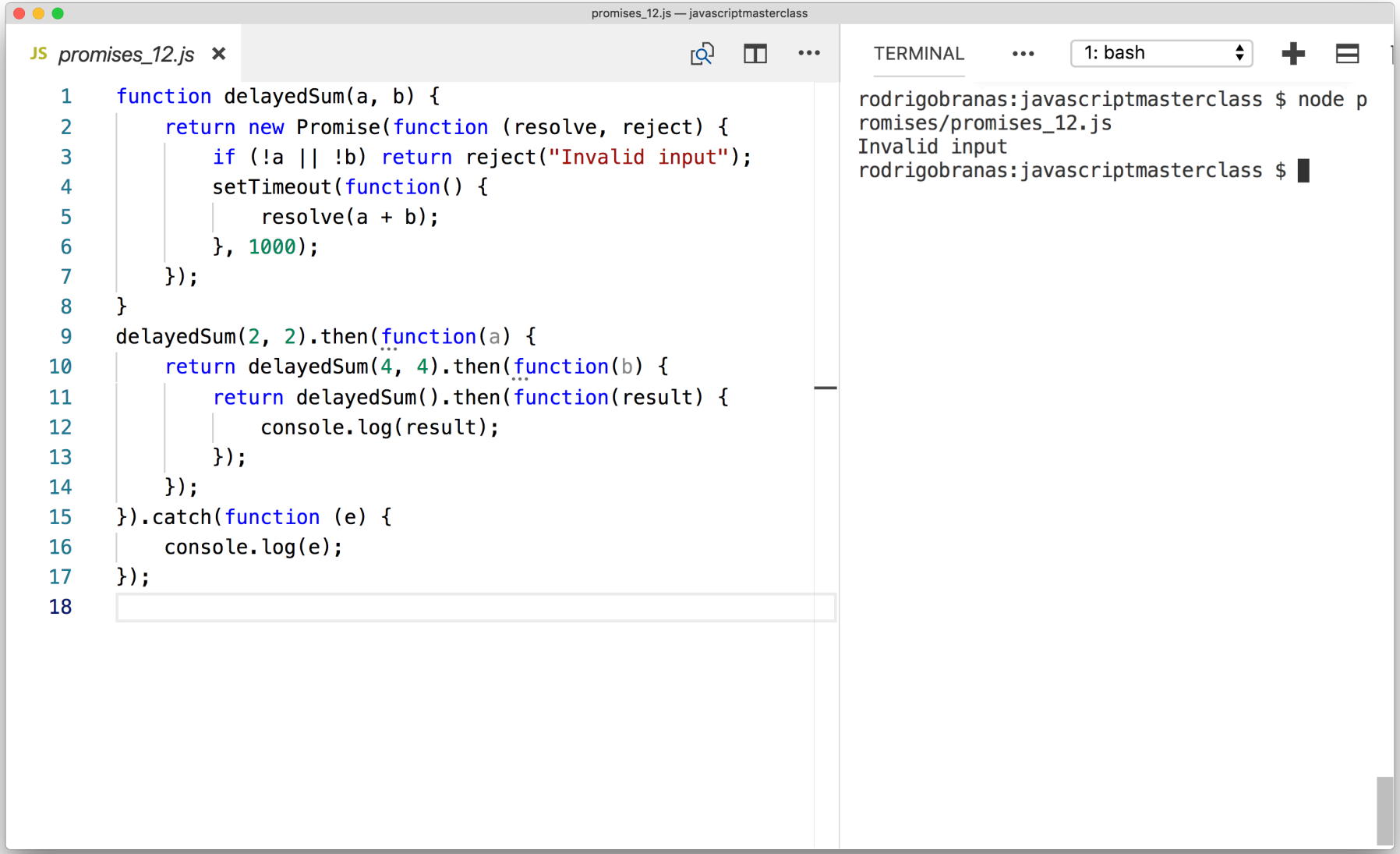

promises_8.js — javascriptmasterclass
promises_8.js
function delayedSum(a, b) {
return new Promise(function (resolve, reject) {
if (!a || !b) return reject("Invalid input");
setTimeout(function() {
resolve(a + b);
}, 1000);
});
}
delayedSum(2, 2).then(function(a) {
delayedSum().then(function(b) {
delayedSum().then(function(result) {
console.log(result);
}).catch(function (e) {
console.log(e);
});
});
});
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node promises/promises_8.js
(node:62191) UnhandledPromiseRejectionWarning: Unhandled promise rejection (rejection id: 2): Invalid input
(node:62191) [DEP0018] DeprecationWarning: Unhandled promise rejections are deprecated. In the future, promise rejections that are not handled will terminate the Node.js process with a non-zero exit code.
rodrigobranas:javascriptmasterclass $

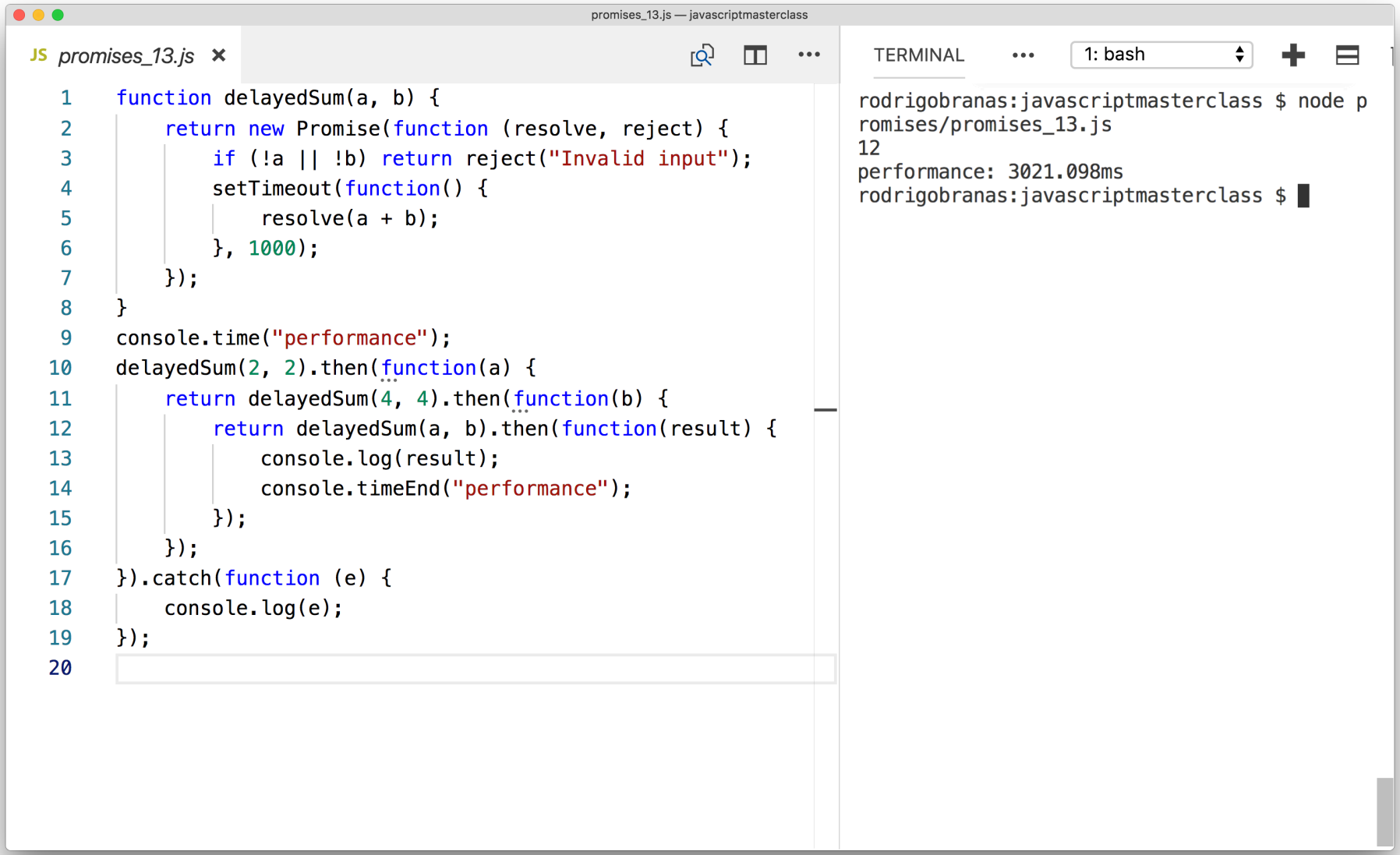

promises_9.js — javascriptmasterclass
promises_9.js
function delayedSum(a, b) {
return new Promise(function (resolve, reject) {
if (!a || !b) return reject("Invalid input");
setTimeout(function() {
resolve(a + b);
}, 1000);
});
}
delayedSum(2, 2).then(function(a) {
delayedSum().then(function(b) {
delayedSum().then(function(result) {
console.log(result);
}).catch(function (e) {
console.log(e);
});
}).catch(function (e) {
console.log(e);
});
});
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
romises/promises_9.js
Invalid input
rodrigobranas:javascriptmasterclass $


promises_10.js — javascriptmasterclass
promises_10.js
function delayedSum(a, b) {
return new Promise(function (resolve, reject) {
if (!a || !b) return reject("Invalid input");
setTimeout(function() {
resolve(a + b);
}, 1000);
});
}
delayedSum().then(function(a) {
delayedSum().then(function(b) {
delayedSum().then(function(result) {
console.log(result);
}).catch(function (e) {
console.log(e);
});
}).catch(function (e) {
console.log(e);
});
});
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node promises/promises_10.js
(node:62223) UnhandledPromiseRejectionWarning: Unhandled promise rejection (rejection id: 2): Invalid input
(node:62223) [DEP0018] DeprecationWarning: Unhandled promise rejections are deprecated. In the future, promise rejections that are not handled will terminate the Node.js process with a non-zero exit code.
rodrigobranas:javascriptmasterclass $

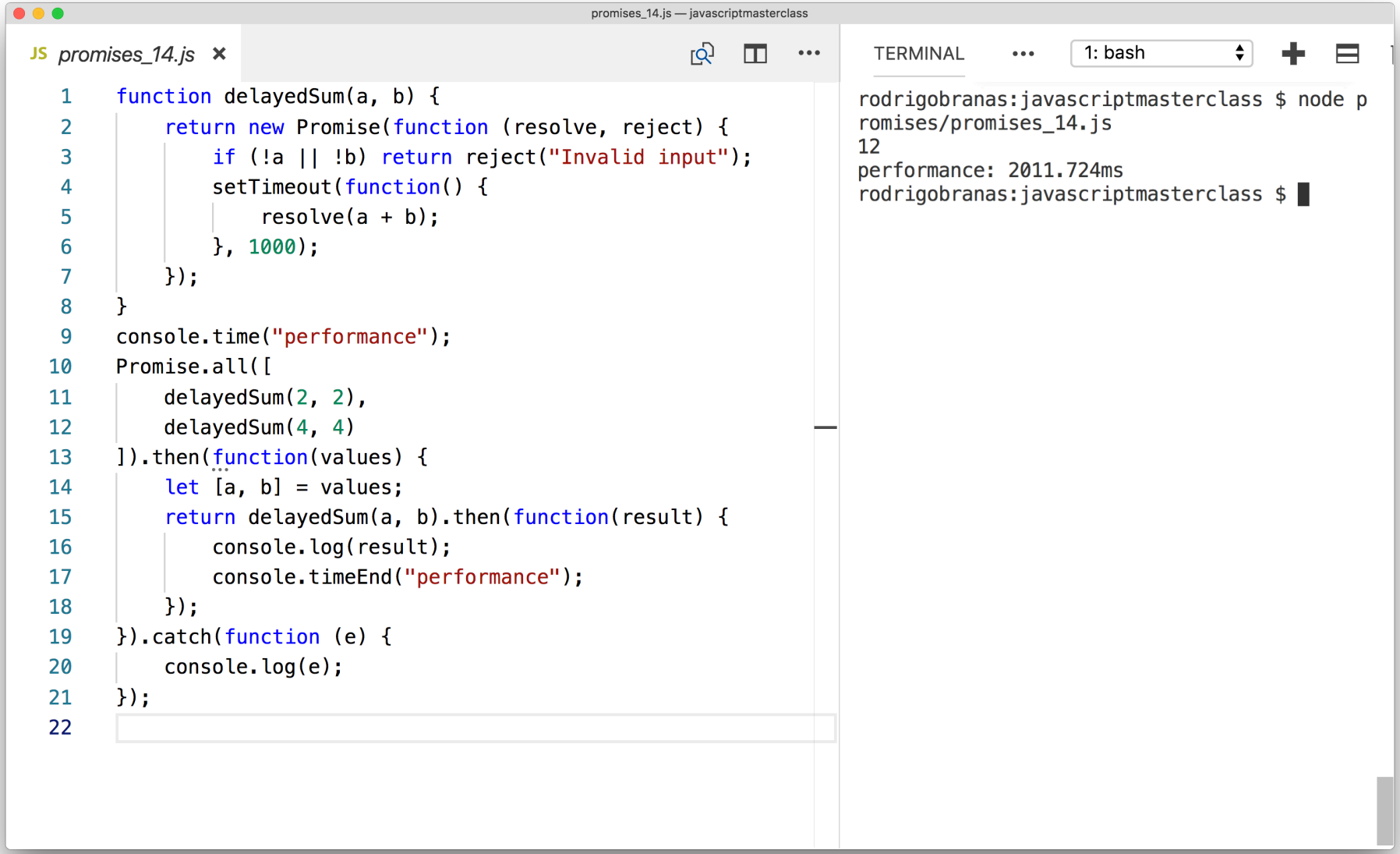

promises_11.js — javascriptmasterclass
promises_11.js
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
romises/promises_11.js
Invalid input
rodrigobranas:javascriptmasterclass $
function delayedSum(a, b) {
return new Promise(function (resolve, reject) {
if (!a || !b) return reject("Invalid input");
setTimeout(function() {
resolve(a + b);
}, 1000);
});
}
delayedSum().then(function(a) {
delayedSum().then(function(b) {
delayedSum().then(function(result) {
console.log(result);
}).catch(function (e) {
console.log(e);
});
}).catch(function (e) {
console.log(e);
});
}).catch(function (e) {
console.log(e);
});

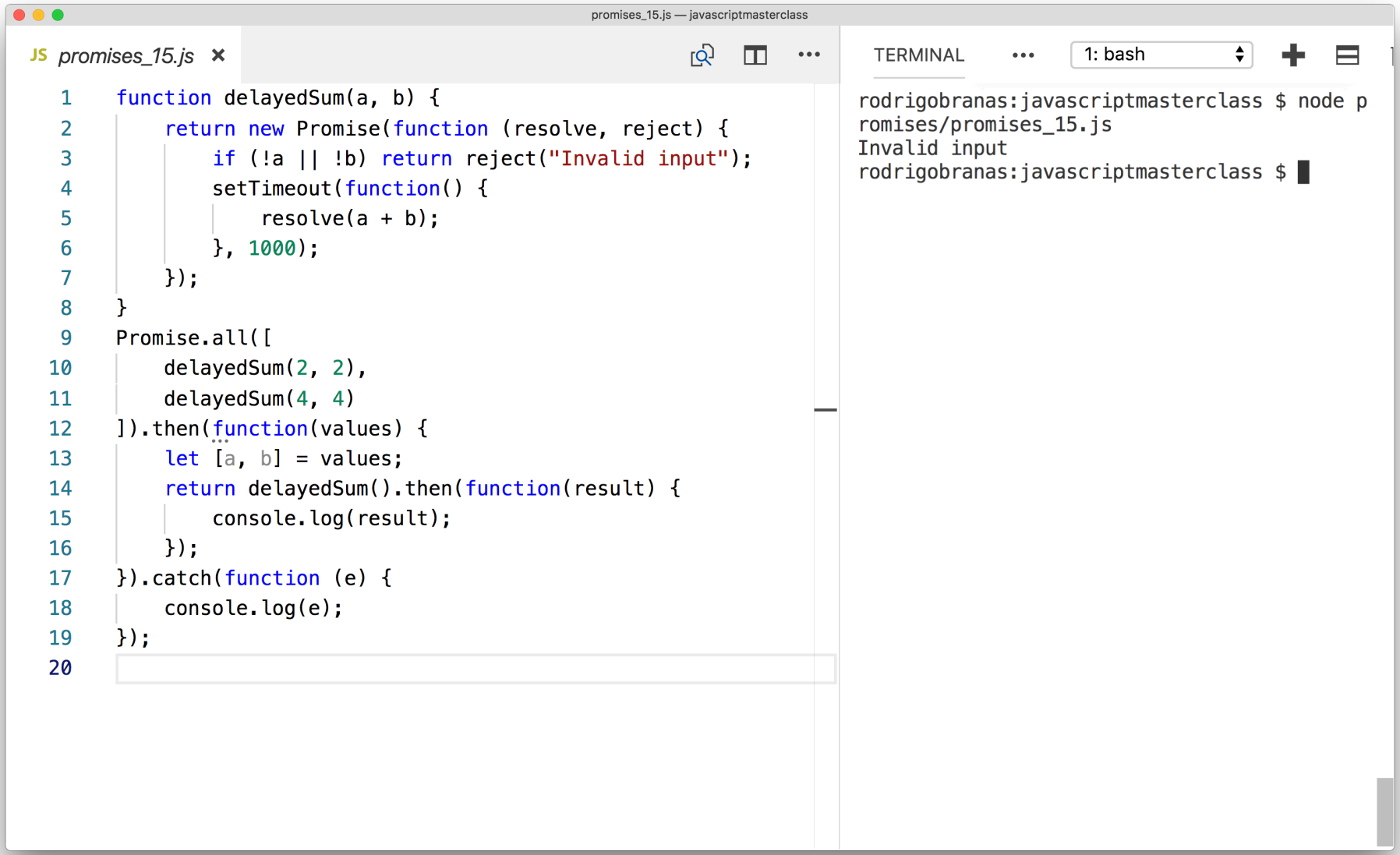
É possível centralizar o tratamento de uma promise encadeando seus retornos

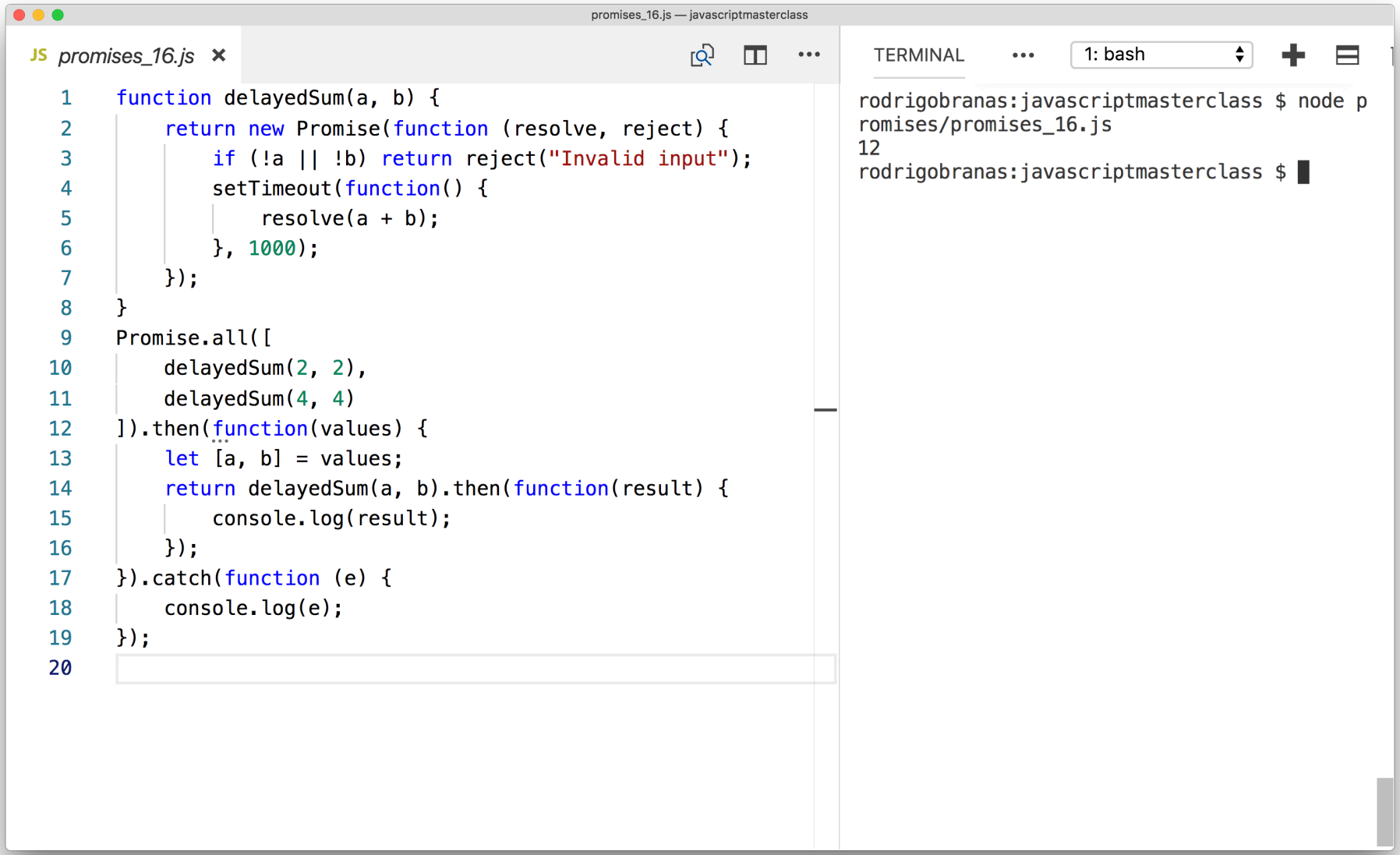

promises_12.js — javascriptmasterclass
promises_12.js
function delayedSum(a, b) {
return new Promise(function (resolve, reject) {
if (!a || !b) return reject("Invalid input");
setTimeout(function() {
resolve(a + b);
}, 1000);
});
}
delayedSum(2, 2).then(function(a) {
return delayedSum(4, 4).then(function(b) {
return delayedSum().then(function(result) {
console.log(result);
});
});
}).catch(function (e) {
console.log(e);
});
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
romises/promises_12.js
Invalid input
rodrigobranas:javascriptmasterclass $


promises_13.js — javascriptmasterclass
promises_13.js
function delayedSum(a, b) {
return new Promise(function (resolve, reject) {
if (!a || !b) return reject("Invalid input");
setTimeout(function() {
resolve(a + b);
}, 1000);
});
}
console.time("performance");
delayedSum(2, 2).then(function(a) {
return delayedSum(4, 4).then(function(b) {
return delayedSum(a, b).then(function(result) {
console.log(result);
console.timeEnd("performance");
});
});
}).catch(function (e) {
console.log(e);
});
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
romises/promises_13.js
12
performance: 3021.098ms
rodrigobranas:javascriptmasterclass $

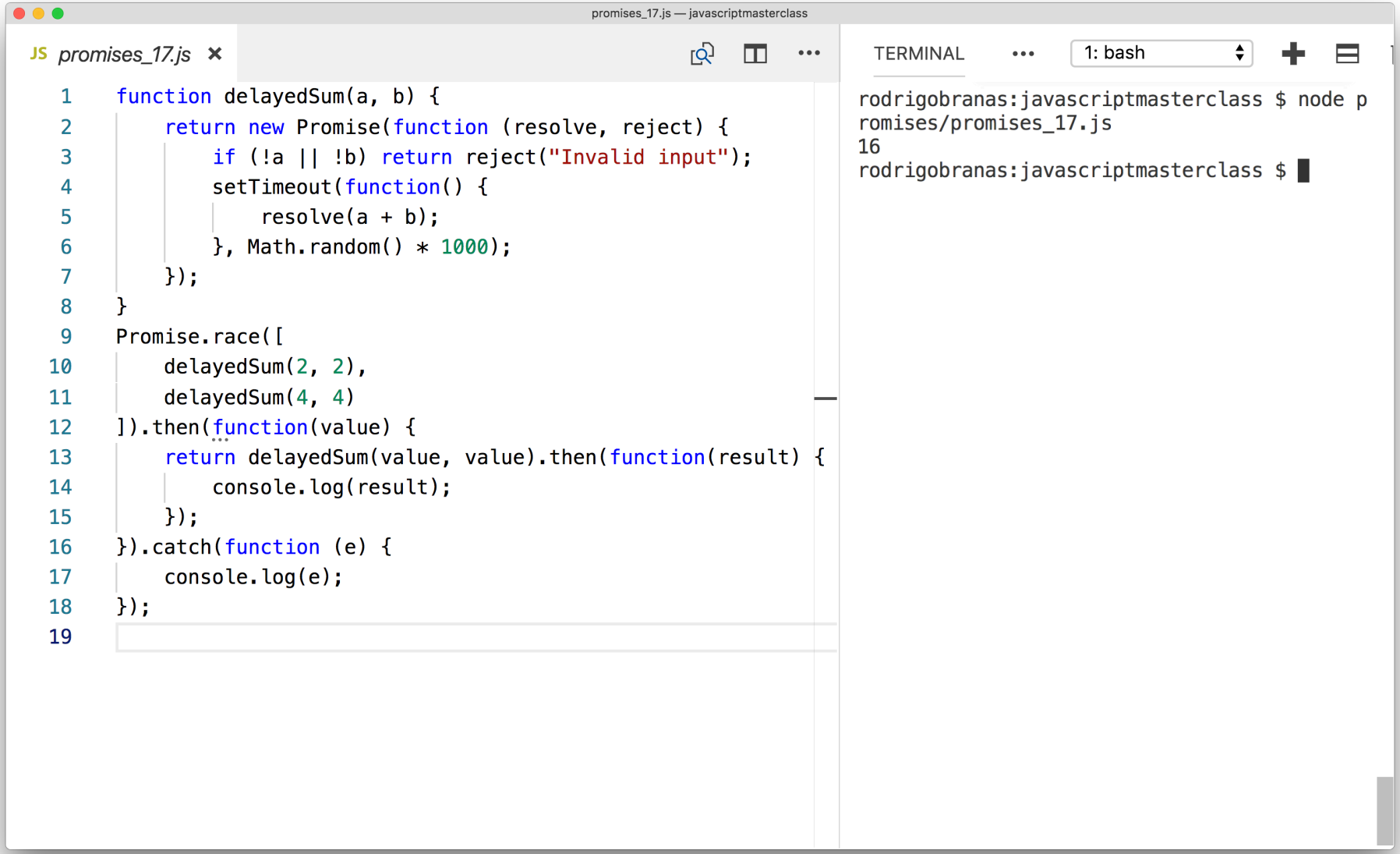
Podemos executar várias promises ao mesmo tempo, retornando após todas terem sucesso usando Promise.all


promises_14.js — javascriptmasterclass
promises_14.js
function delayedSum(a, b) {
return new Promise(function (resolve, reject) {
if (!a || !b) return reject("Invalid input");
setTimeout(function() {
resolve(a + b);
}, 1000);
});
}
console.time("performance");
Promise.all([
delayedSum(2, 2),
delayedSum(4, 4)
]).then(function(values) {
let [a, b] = values;
return delayedSum(a, b).then(function(result) {
console.log(result);
console.timeEnd("performance");
});
}).catch(function (e) {
console.log(e);
});
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
romises/promises_14.js
12
performance: 2011.724ms
rodrigobranas:javascriptmasterclass $


promises_15.js — javascriptmasterclass
promises_15.js
TERMINAL
1: bash
function delayedSum(a, b) {
return new Promise(function (resolve, reject) {
if (!a || !b) return reject("Invalid input");
setTimeout(function() {
resolve(a + b);
}, 1000);
});
}
Promise.all([
delayedSum(2, 2),
delayedSum(4, 4)
]).then(function(values) {
let [a, b] = values;
return delayedSum().then(function(result) {
console.log(result);
});
}).catch(function (e) {
console.log(e);
});
rodrigobranas:javascriptmasterclass $ node p
romises/promises_15.js
Invalid input
rodrigobranas:javascriptmasterclass $


promises_16.js — javascriptmasterclass
promises_16.js
function delayedSum(a, b) {
return new Promise(function (resolve, reject) {
if (!a || !b) return reject("Invalid input");
setTimeout(function() {
resolve(a + b);
}, 1000);
});
}
Promise.all([
delayedSum(2, 2),
delayedSum(4, 4)
]).then(function(values) {
let [a, b] = values;
return delayedSum(a, b).then(function(result) {
console.log(result);
});
}).catch(function (e) {
console.log(e);
});
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
romises/promises_16.js
12
rodrigobranas:javascriptmasterclass $

Também podemos executar várias promises ao mesmo, retornando após a primeira ter sucesso usando Promise.race


promises_17.js — javascriptmasterclass
promises_17.js
function delayedSum(a, b) {
return new Promise(function (resolve, reject) {
if (!a || !b) return reject("Invalid input");
setTimeout(function() {
resolve(a + b);
}, Math.random() * 1000);
});
}
Promise.race([
delayedSum(2, 2),
delayedSum(4, 4)
]).then(function(value) {
return delayedSum(value, value).then(function(result) {
console.log(result);
});
}).catch(function (e) {
console.log(e);
});
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node p
romises/promises_17.js
16
rodrigobranas:javascriptmasterclass $