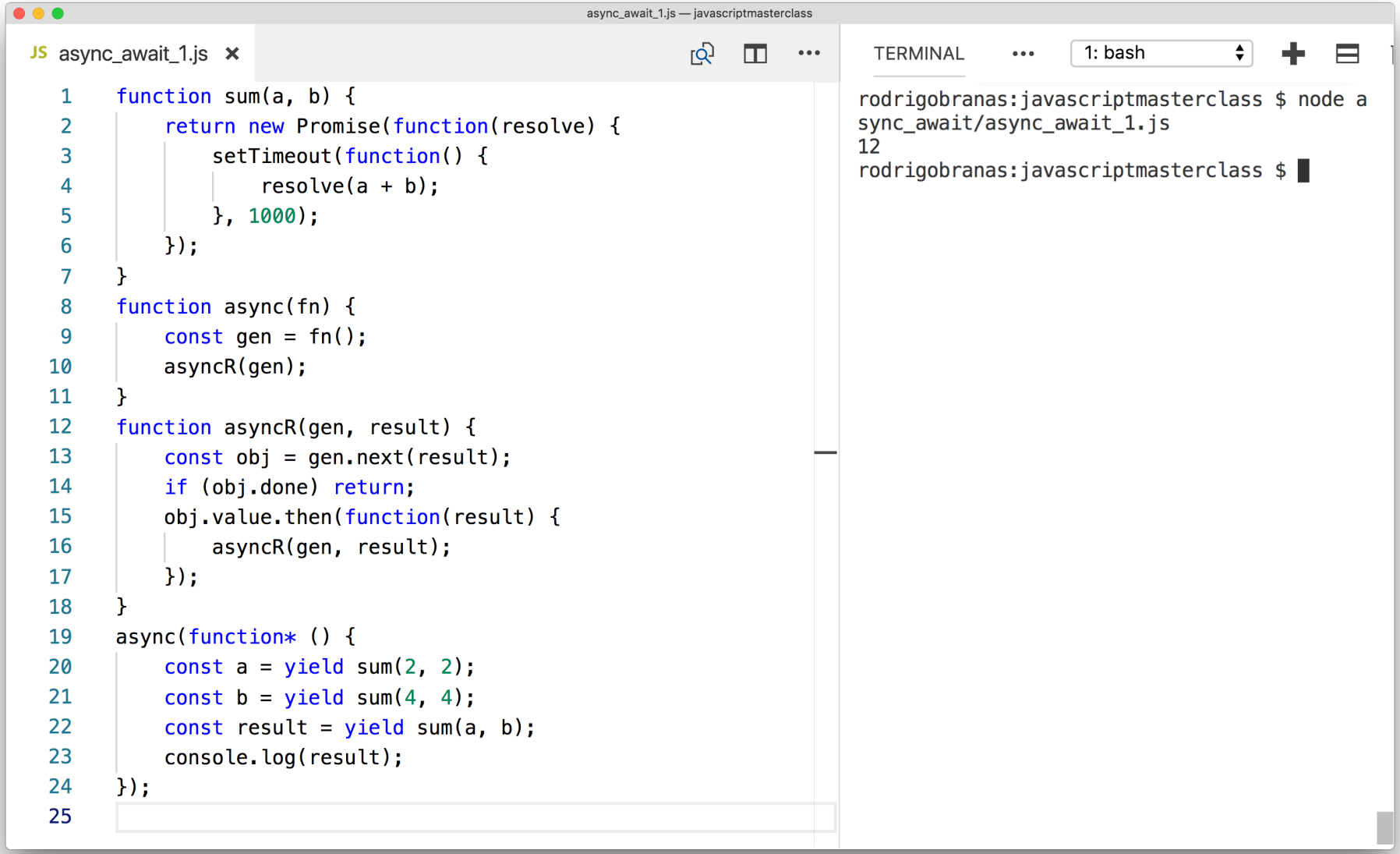
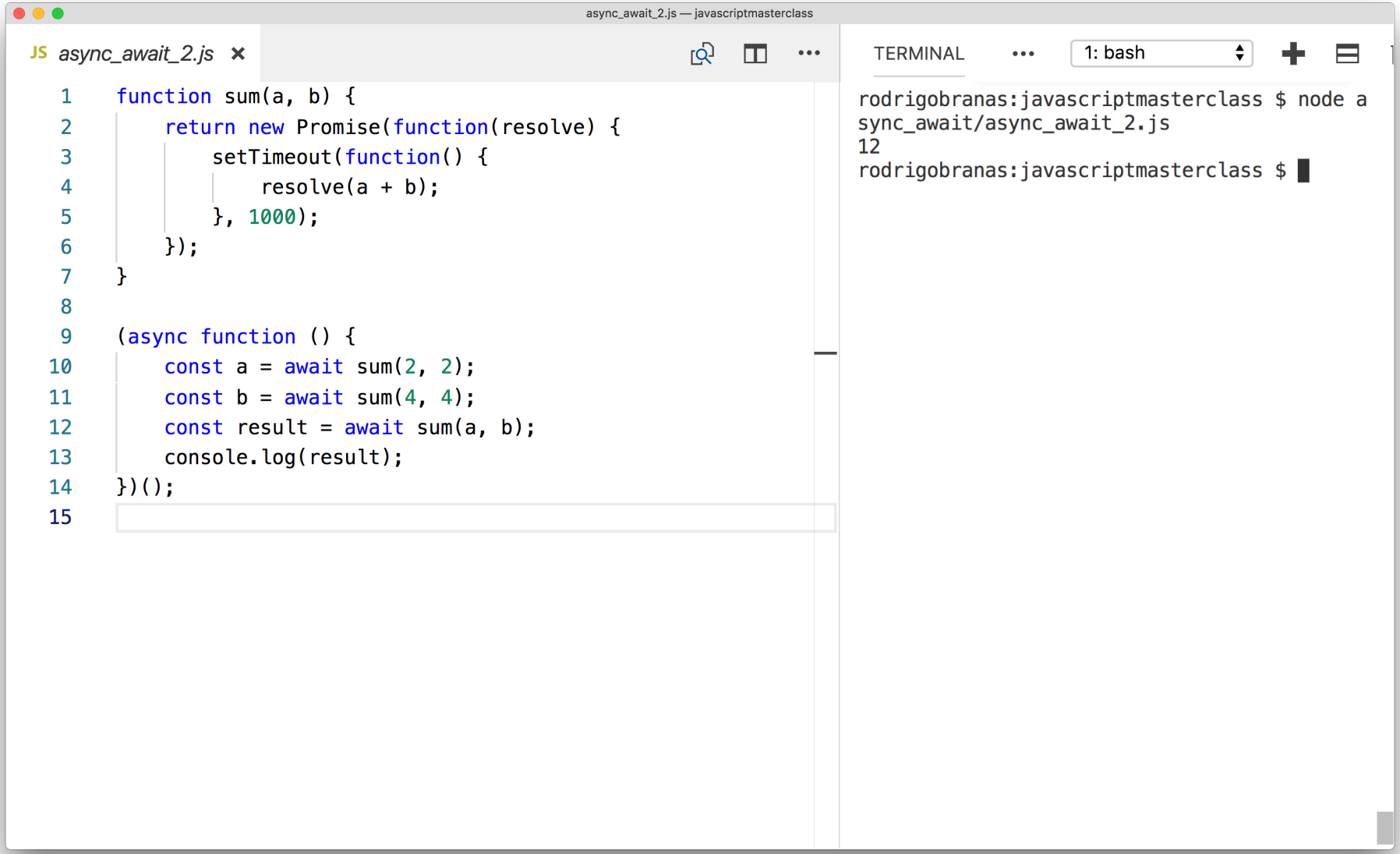
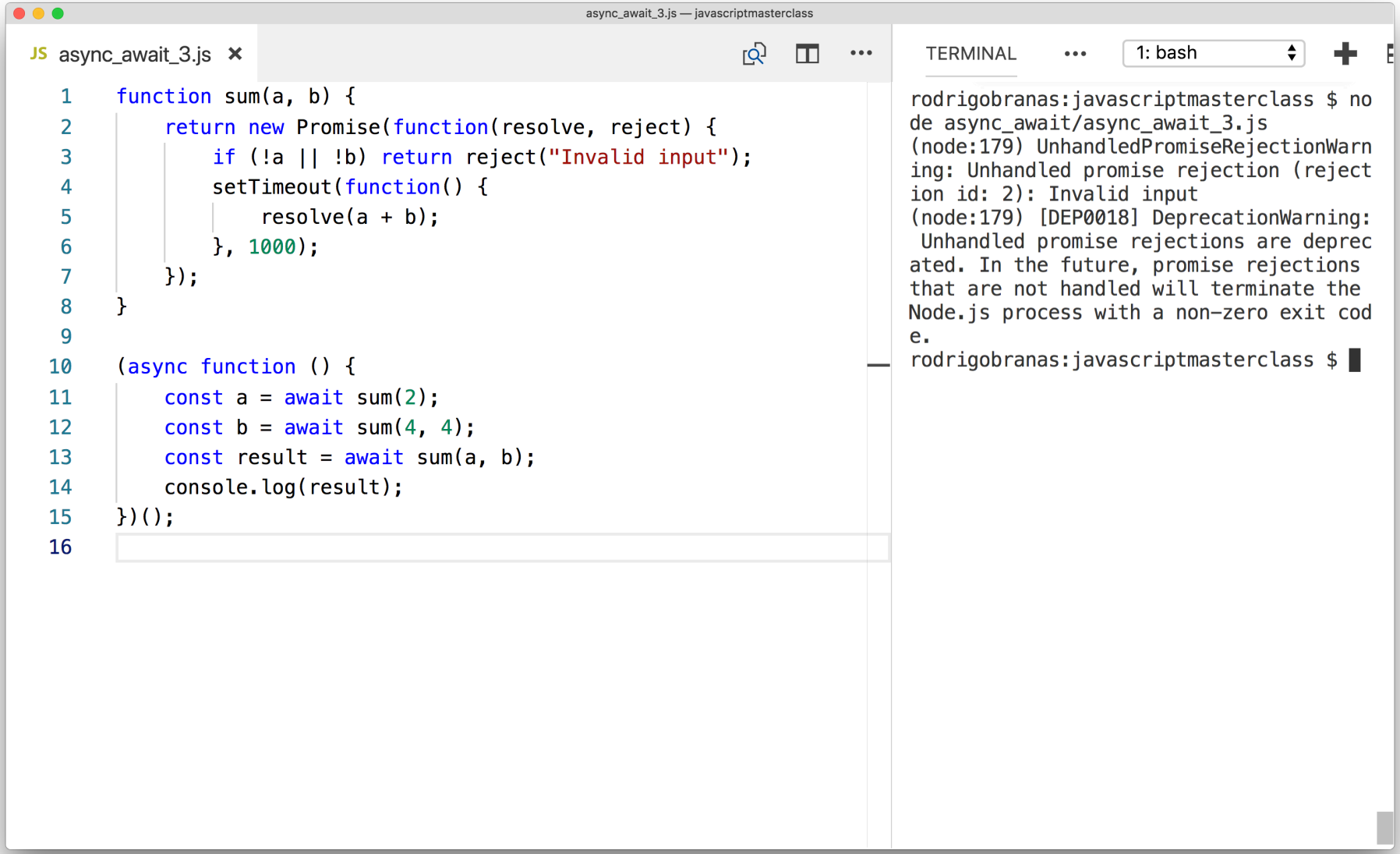
Async/Await

O async/await facilita a interação com chamadas assíncronas, aguardando o retorno de uma determinada promise

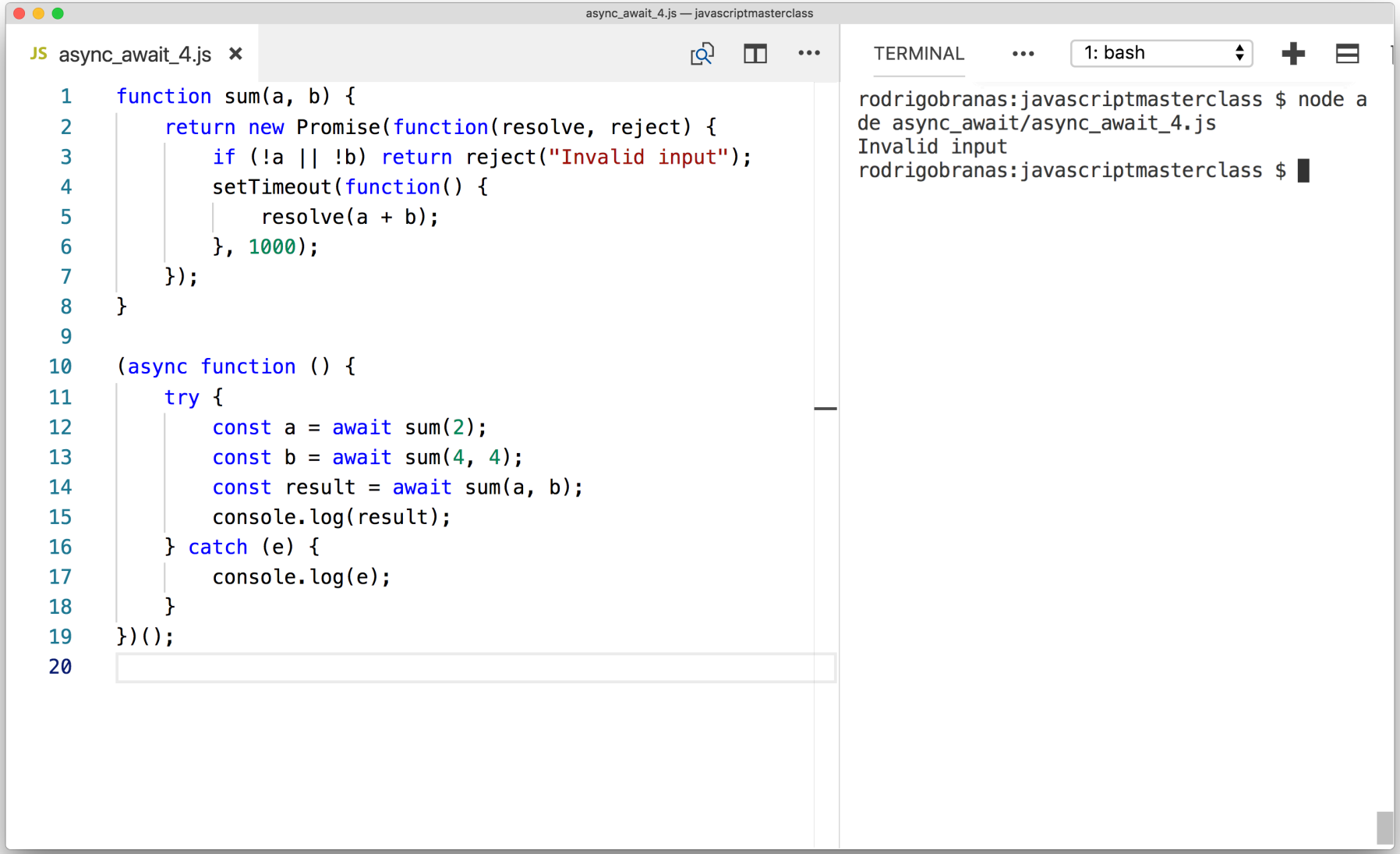


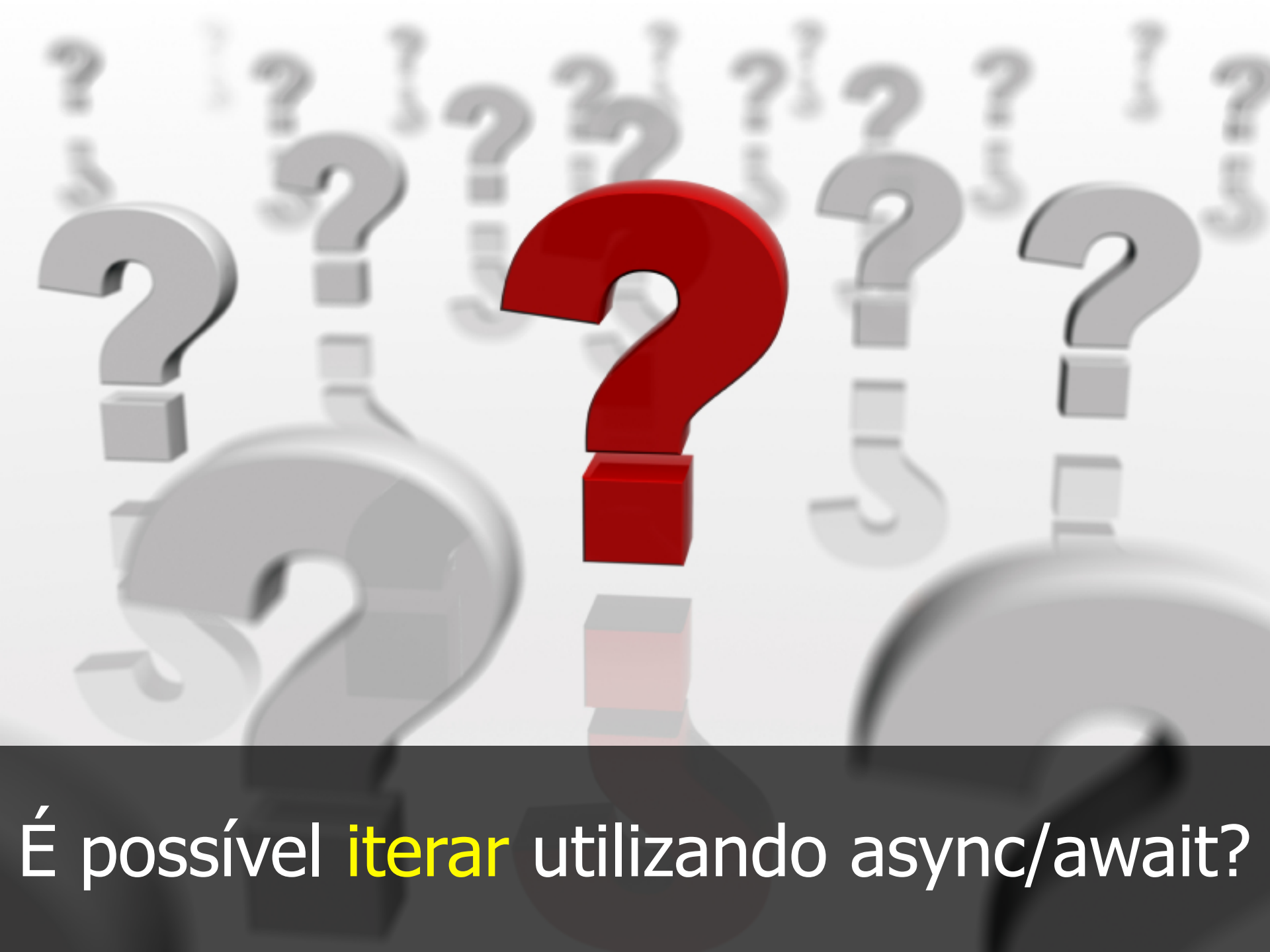

```
function sum(a, b) {
    return new Promise(function(resolve) {
        setTimeout(function() {
            resolve(a + b);
        }, 1000);
    });
}
function async(fn) {
    const gen = fn();
    asyncR(gen);
}
function asyncR(gen, result) {
    const obj = gen.next(result);
    if (obj.done) return;
    obj.value.then(function(result) {
        asyncR(gen, result);
    });
}
async(function* () {
    const a = yield sum(2, 2);
    const b = yield sum(4, 4);
    const result = yield sum(a, b);
    console.log(result);
});
```


async_await_2.js — javascriptmasterclass
async_await_2.js
1 function sum(a, b) {
2 return new Promise(function(resolve) {
3 setTimeout(function() {
4 resolve(a + b);
5 }, 1000);
6 });
7 }
8
9 (async function () {
10 const a = await sum(2, 2);
11 const b = await sum(4, 4);
12 const result = await sum(a, b);
13 console.log(result);
14 })();
15
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node a
sync_await/async_await_2.js
12
rodrigobranas:javascriptmasterclass $

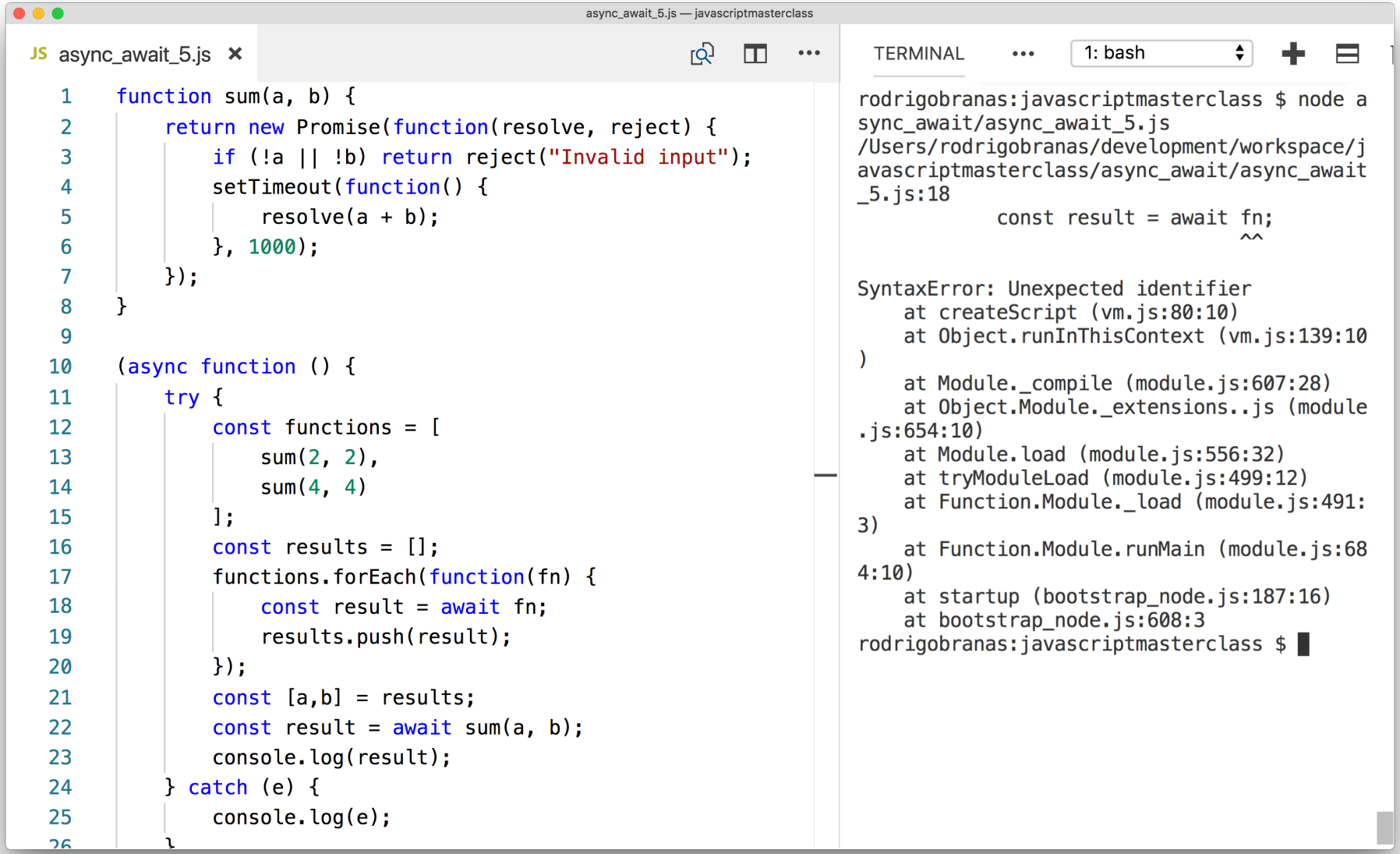

async_await_3.js — javascriptmasterclass
async_await_3.js
function sum(a, b) {
return new Promise(function(resolve, reject) {
if (!a || !b) return reject("Invalid input");
setTimeout(function() {
resolve(a + b);
}, 1000);
});
}
(async function () {
const a = await sum(2);
const b = await sum(4, 4);
const result = await sum(a, b);
console.log(result);
})();
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node async_await/async_await_3.js
(node:179) UnhandledPromiseRejectionWarning: Unhandled promise rejection (rejection id: 2): Invalid input
(node:179) [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 $

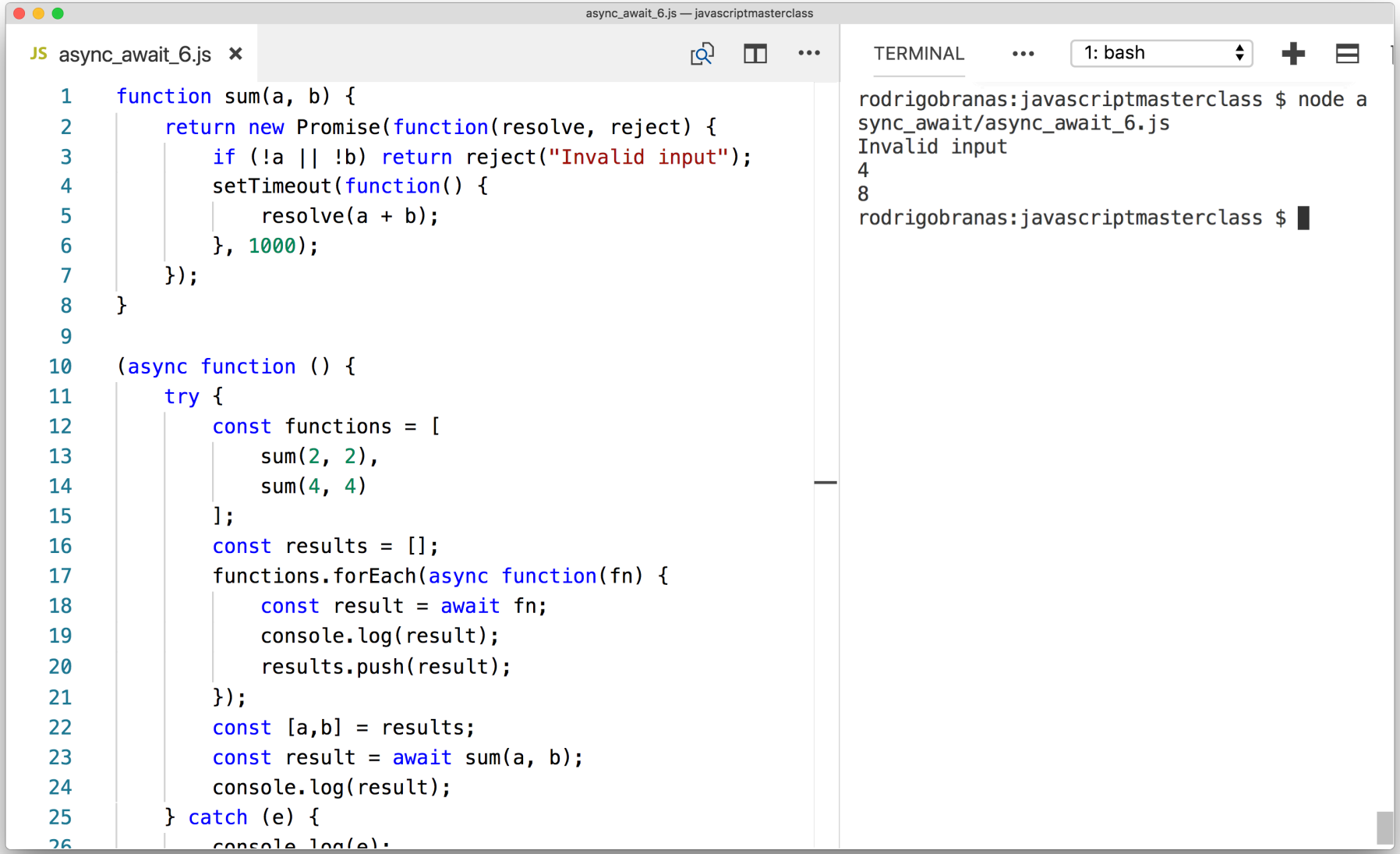
Para tratar possíveis exceções associadas a chamadas assíncronas é possível utilizar um bloco try/catch

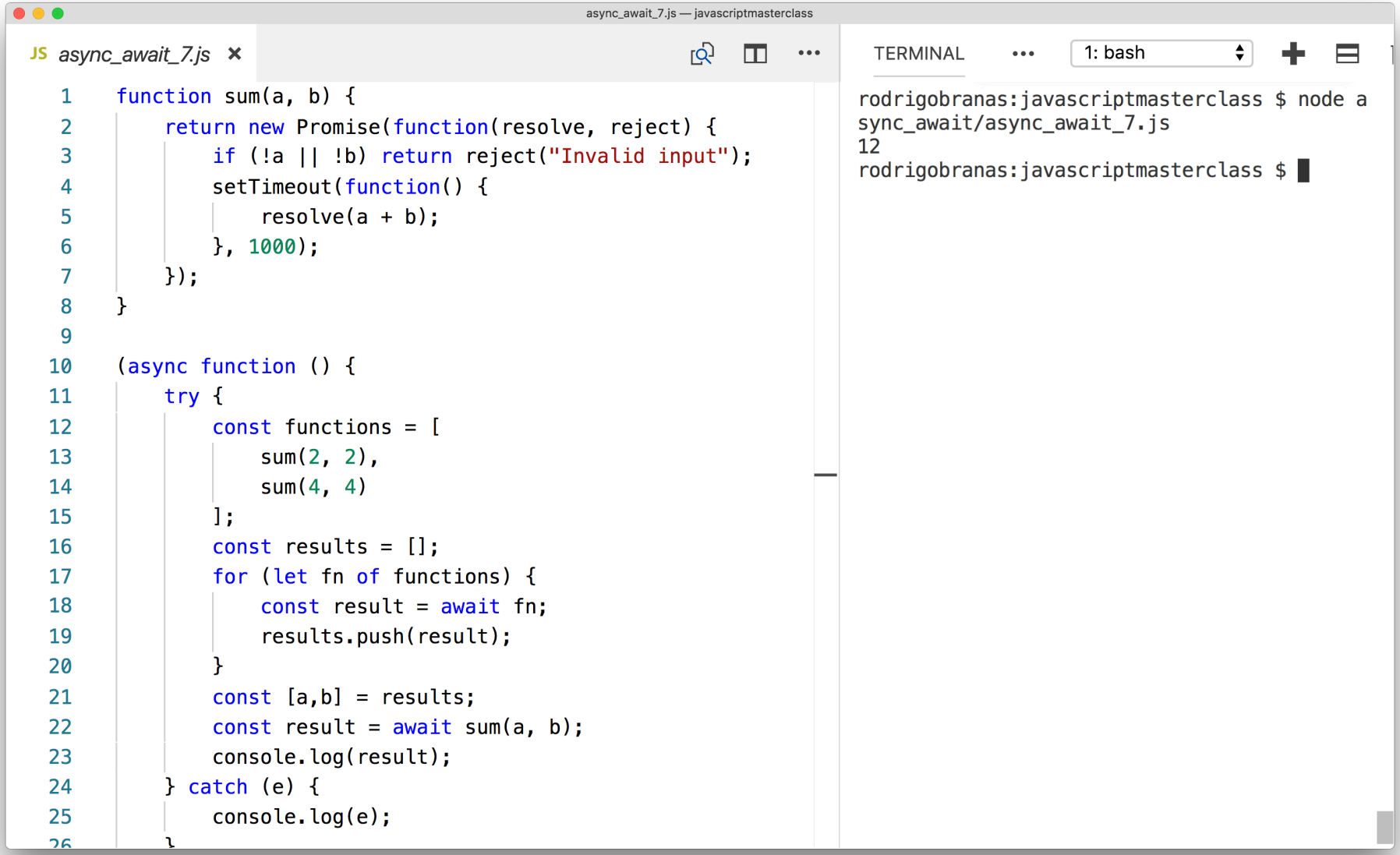

async_await_4.js — javascriptmasterclass
async_await_4.js
function sum(a, b) {
return new Promise(function(resolve, reject) {
if (!a || !b) return reject("Invalid input");
setTimeout(function() {
resolve(a + b);
}, 1000);
});
}
(async function () {
try {
const a = await sum(2);
const b = await sum(4, 4);
const result = await sum(a, b);
console.log(result);
} catch (e) {
console.log(e);
}
})();
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node a
de async_await/async_await_4.js
Invalid input
rodrigobranas:javascriptmasterclass $



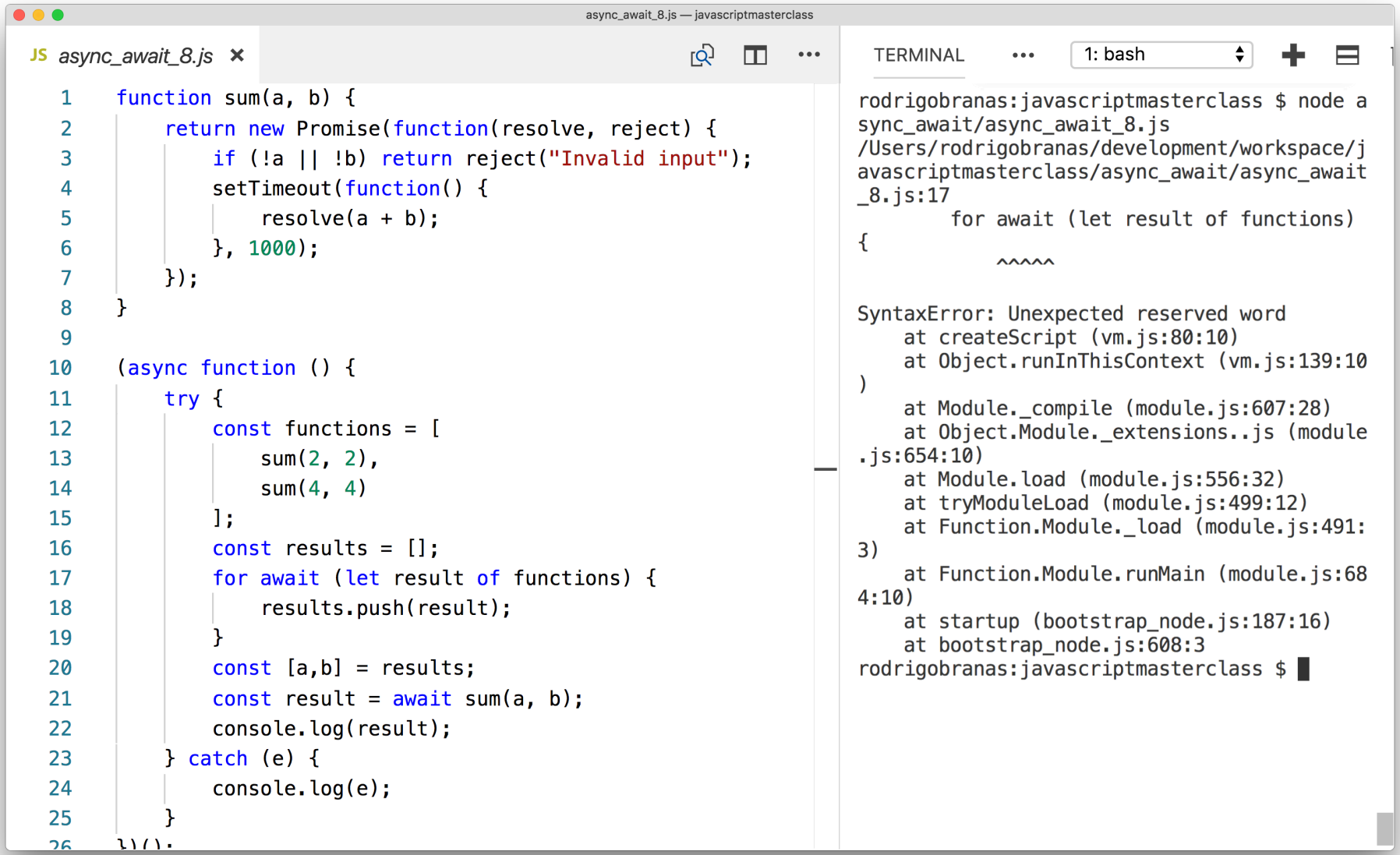
É possível iterar utilizando async/await?


async_await_5.js — javascriptmasterclass
async_await_5.js
function sum(a, b) {
return new Promise(function(resolve, reject) {
if (!a || !b) return reject("Invalid input");
setTimeout(function() {
resolve(a + b);
}, 1000);
});
}
(async function () {
try {
const functions = [
sum(2, 2),
sum(4, 4)
];
const results = [];
functions.forEach(function(fn) {
const result = await fn;
results.push(result);
});
const [a,b] = results;
const result = await sum(a, b);
console.log(result);
} catch (e) {
console.log(e);
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node async_await/async_await_5.js
/Users/rodrigobranas/development/workspace/javascriptmasterclass/async_await/async_await_5.js:18
const result = await fn;
^^
SyntaxError: Unexpected identifier
at createScript (vm.js:80:10)
at Object.runInThisContext (vm.js:139:10)
at Module._compile (module.js:607:28)
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 $

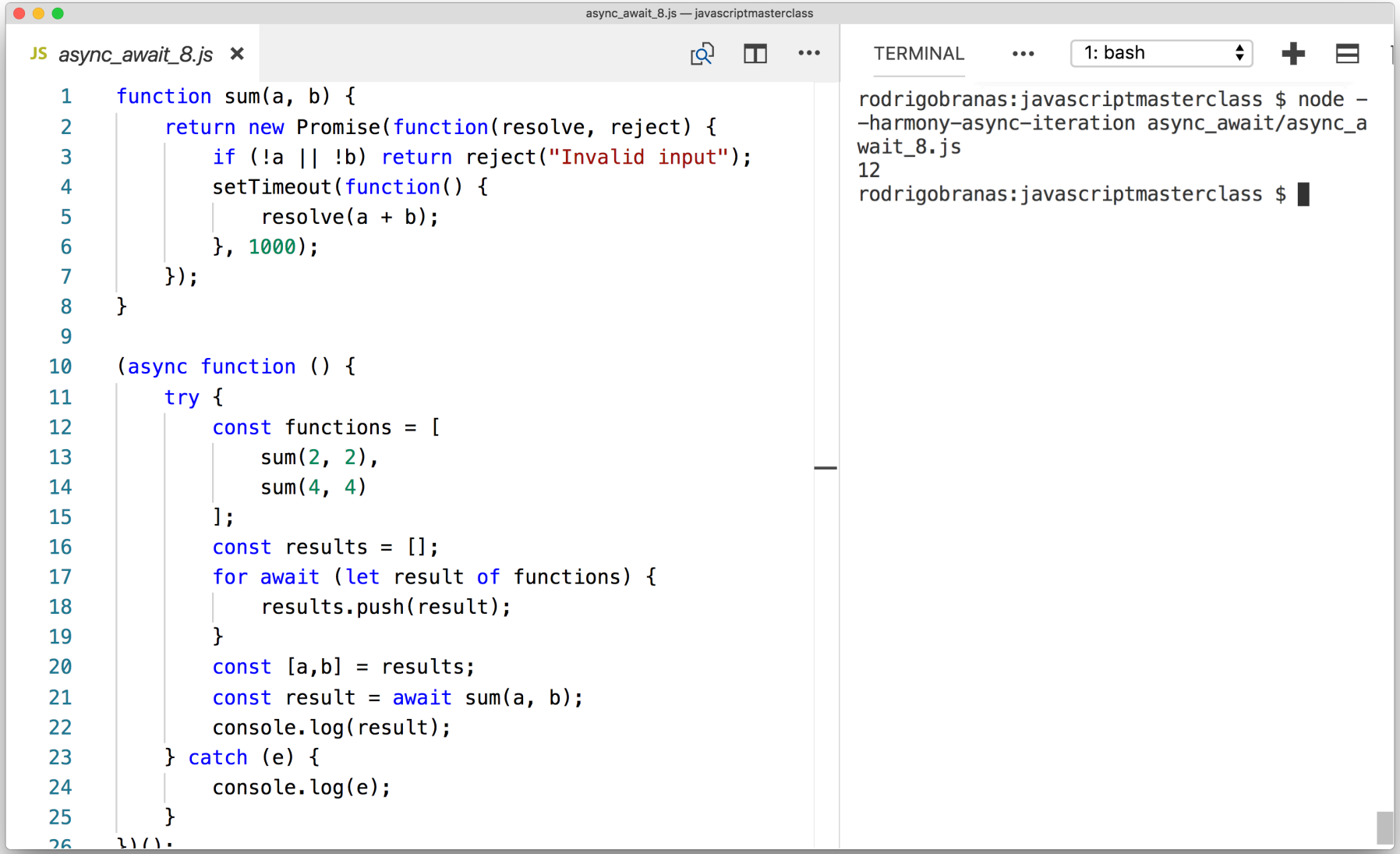

async_await_6.js — javascriptmasterclass
async_await_6.js
TERMINAL
1: bash
function sum(a, b) {
return new Promise(function(resolve, reject) {
if (!a || !b) return reject("Invalid input");
setTimeout(function() {
resolve(a + b);
}, 1000);
});
}
(async function () {
try {
const functions = [
sum(2, 2),
sum(4, 4)
];
const results = [];
functions.forEach(async function(fn) {
const result = await fn;
console.log(result);
results.push(result);
});
const [a,b] = results;
const result = await sum(a, b);
console.log(result);
} catch (e) {
rodrigobranas:javascriptmasterclass $ node a
sync_await/async_await_6.js
Invalid input
4
8
rodrigobranas:javascriptmasterclass $


async_await_7.js — javascriptmasterclass
async_await_7.js
function sum(a, b) {
return new Promise(function(resolve, reject) {
if (!a || !b) return reject("Invalid input");
setTimeout(function() {
resolve(a + b);
}, 1000);
});
}
(async function () {
try {
const functions = [
sum(2, 2),
sum(4, 4)
];
const results = [];
for (let fn of functions) {
const result = await fn;
results.push(result);
}
const [a,b] = results;
const result = await sum(a, b);
console.log(result);
} catch (e) {
console.log(e);
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node a
sync_await/async_await_7.js
12
rodrigobranas:javascriptmasterclass $

É possível utilizar o bloco for-await-of para iterar sobre um iterator de promises


async_await_8.js — javascriptmasterclass
async_await_8.js
function sum(a, b) {
return new Promise(function(resolve, reject) {
if (!a || !b) return reject("Invalid input");
setTimeout(function() {
resolve(a + b);
}, 1000);
});
}
(async function () {
try {
const functions = [
sum(2, 2),
sum(4, 4)
];
const results = [];
for await (let result of functions) {
results.push(result);
}
const [a,b] = results;
const result = await sum(a, b);
console.log(result);
} catch (e) {
console.log(e);
}
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node async_await/async_await_8.js
/Users/rodrigobranas/development/workspace/javascriptmasterclass/async_await/async_await_8.js:17
for await (let result of functions) {
^^^^^
SyntaxError: Unexpected reserved word
at createScript (vm.js:80:10)
at Object.runInThisContext (vm.js:139:10)
at Module._compile (module.js:607:28)
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 $

Para utilizar é necessário usar a flag --harmony-async-iteration


async_await_8.js — javascriptmasterclass
async_await_8.js
function sum(a, b) {
return new Promise(function(resolve, reject) {
if (!a || !b) return reject("Invalid input");
setTimeout(function() {
resolve(a + b);
}, 1000);
});
}
(async function () {
try {
const functions = [
sum(2, 2),
sum(4, 4)
];
const results = [];
for await (let result of functions) {
results.push(result);
}
const [a,b] = results;
const result = await sum(a, b);
console.log(result);
} catch (e) {
console.log(e);
}
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node --harmony-async-iteration async_await/async_await_8.js
12
rodrigobranas:javascriptmasterclass $