Módulos


CommonJS

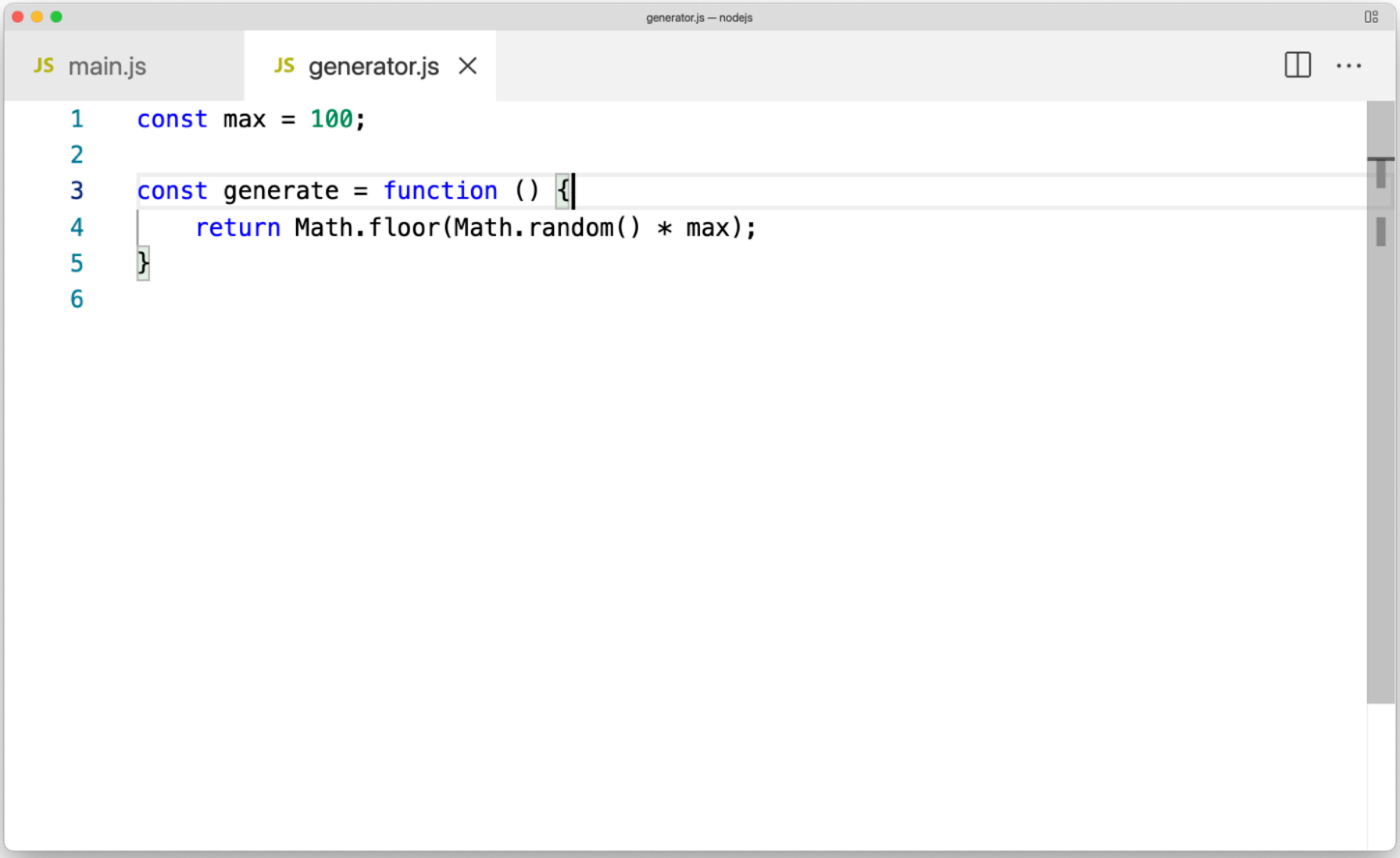
A linguagem JavaScript nasceu dentro do navegador e uma das características da Web era justamente o conceito de Code on Demand e isso era realizado pela tag <SCRIPT>

Como a linguagem oficialmente não tinha suporte a qualquer sistema de módulos, algumas abordagens foram criadas, entre elas o CommonJS ou CJS

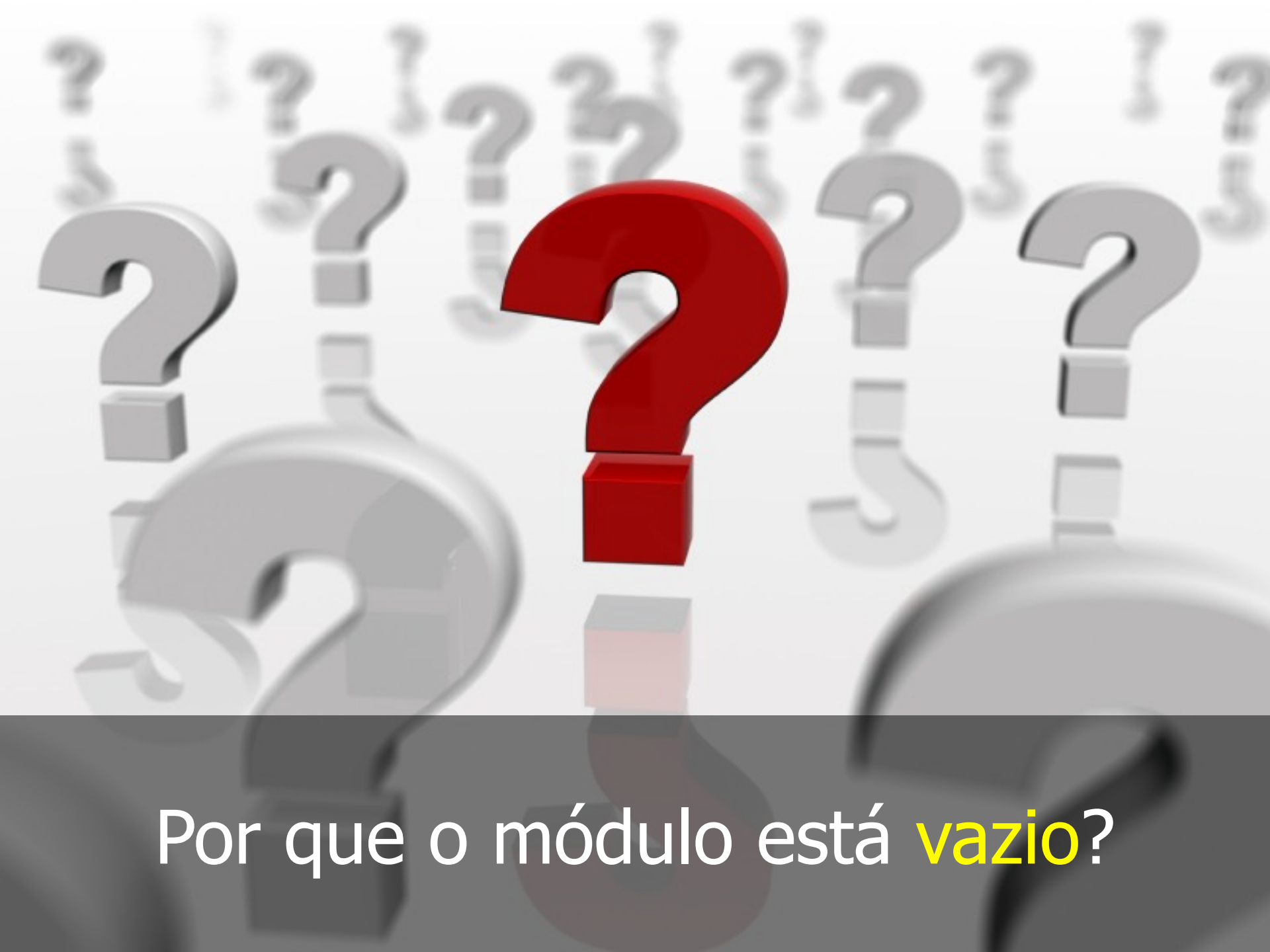
Ele nasceu na Mozilla em 2009 com o nome de ServerJS, sendo renomeado alguns meses depois para CommonJS

Nele, existe uma relação direta entre um módulo de um arquivo, que pode ter parte do seu conteúdo exportado e importado em outro módulo

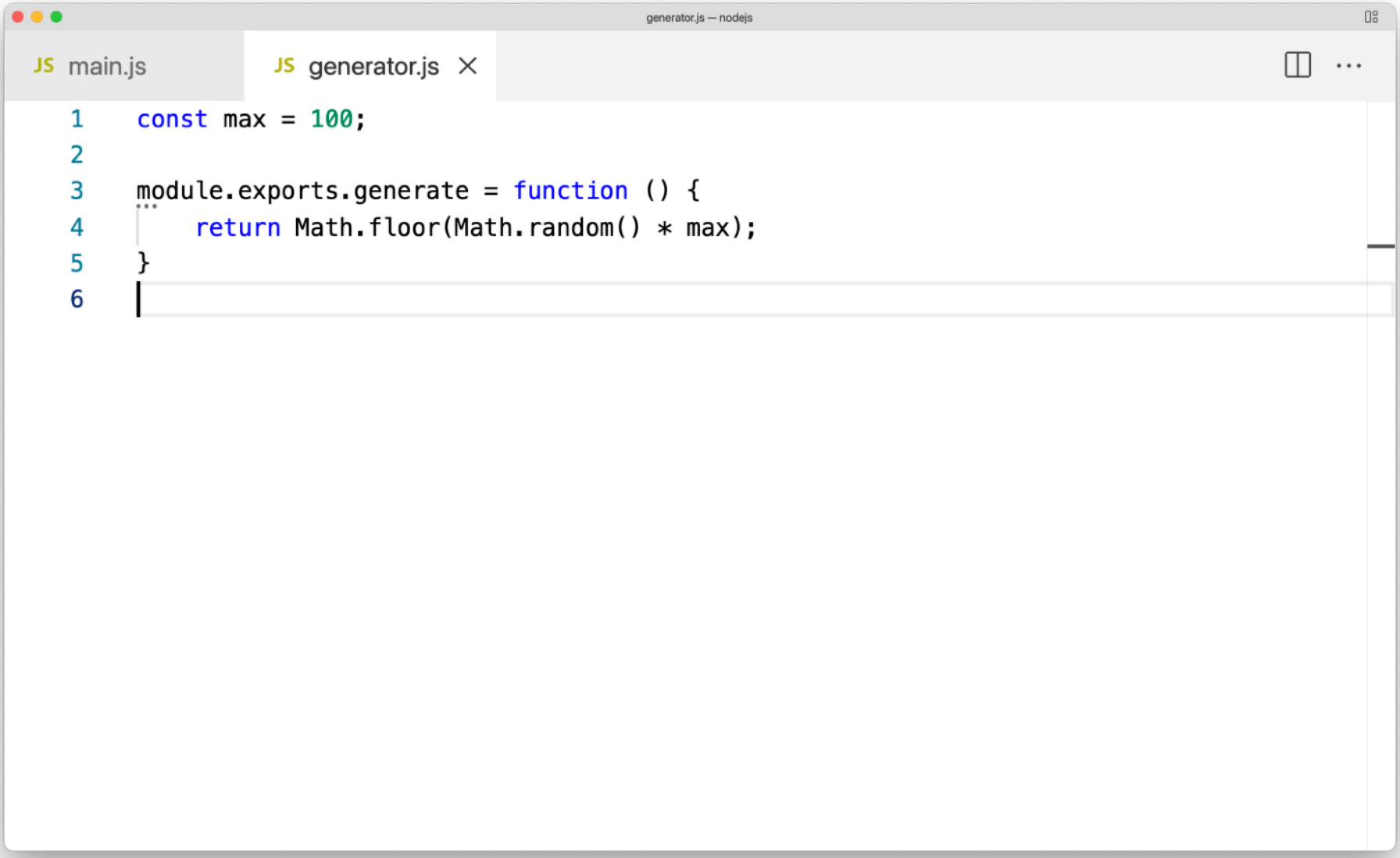
Basicamente é só usar module.exports para exportar e require para importar


generator.js — nodejs
main.js
generator.js
1 const max = 100;
2
3 const generate = function () {
4 return Math.floor(Math.random() * max);
5 }
6


main.js — nodejs
main.js
generator.js
1 const generator = require("./generator");
2 console.log(generator);
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
zsh
rodrigobranas nodejs $ node main.js
{}
rodrigobranas nodejs $


Por que o módulo está vazio?

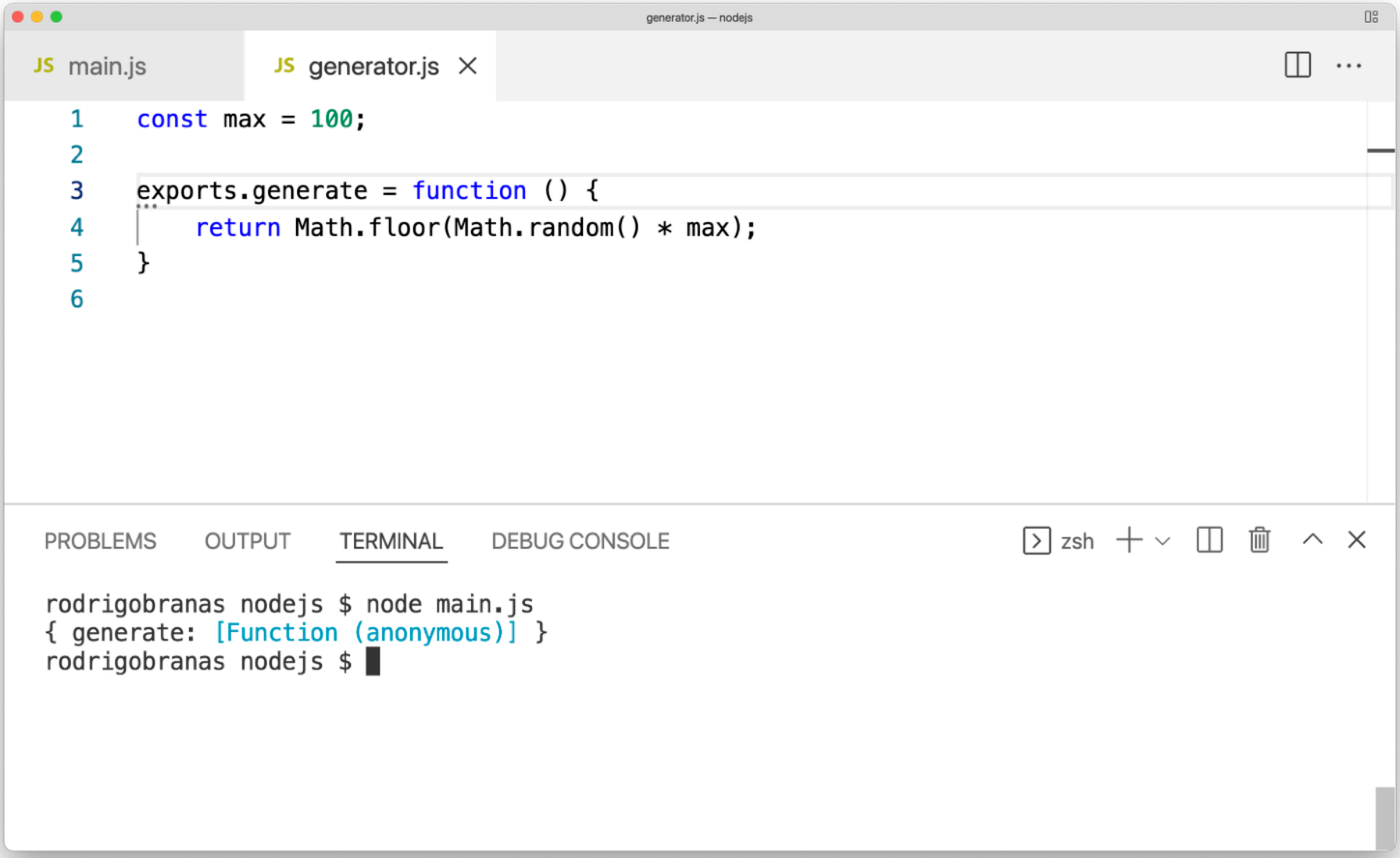
Tudo que é definido dentro do módulo é privado

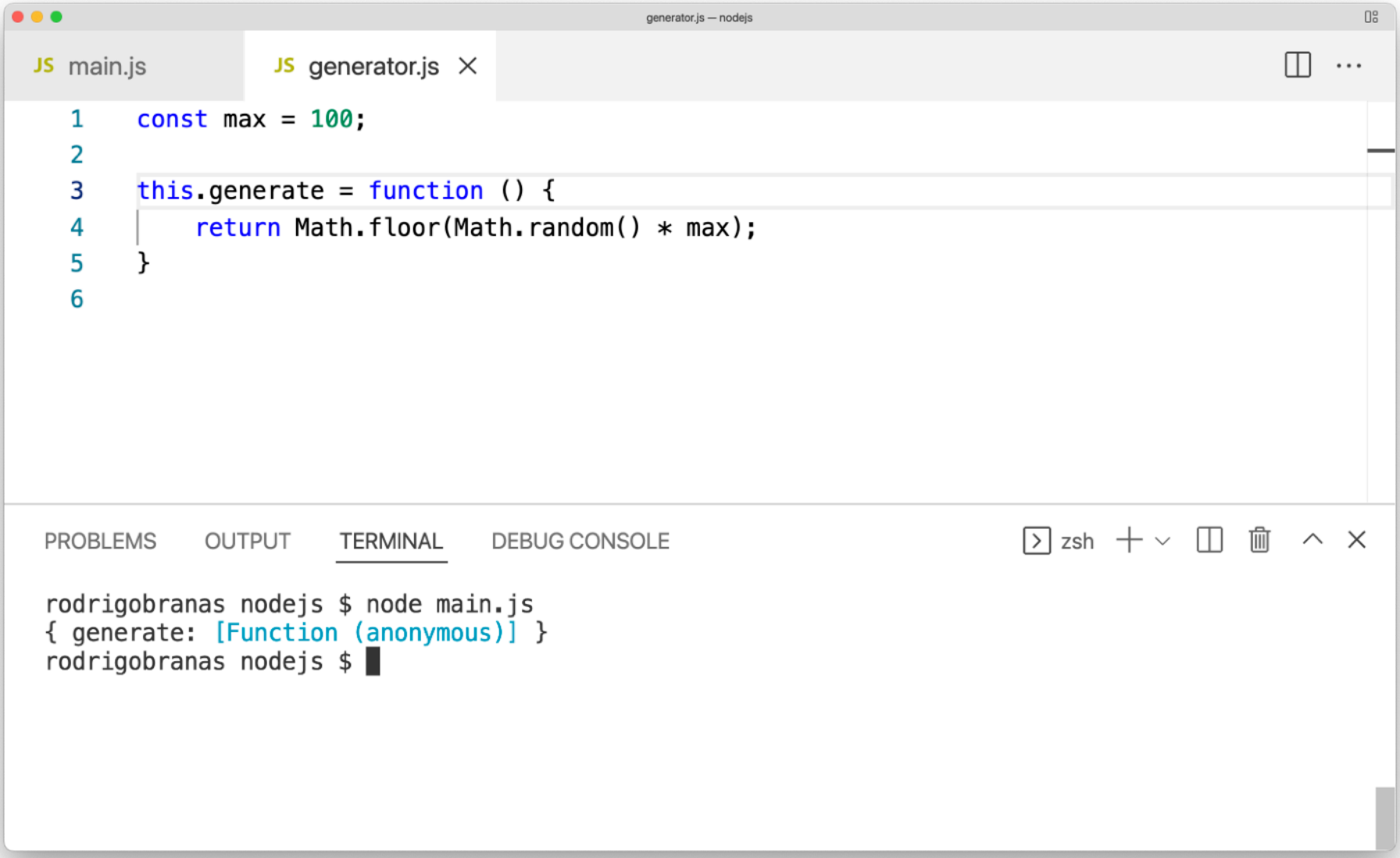

generator.js — nodejs
main.js
generator.js
const max = 100;
module.exports.generate = function () {
return Math.floor(Math.random() * max);
}


main.js — nodejs
main.js
generator.js
1 const generator = require("./generator");
2 console.log(generator);
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
zsh
rodrigobranas nodejs $ node main.js
{ generate: [Function (anonymous)] }
rodrigobranas nodejs $


main.js — nodejs
main.js
generator.js
1 const generator = require("./generator");
2 console.log(generator.generate());
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
zsh
rodrigobranas nodejs $ node main.js
98
rodrigobranas nodejs $

Também é possível exportar o módulo utilizando exports ou this

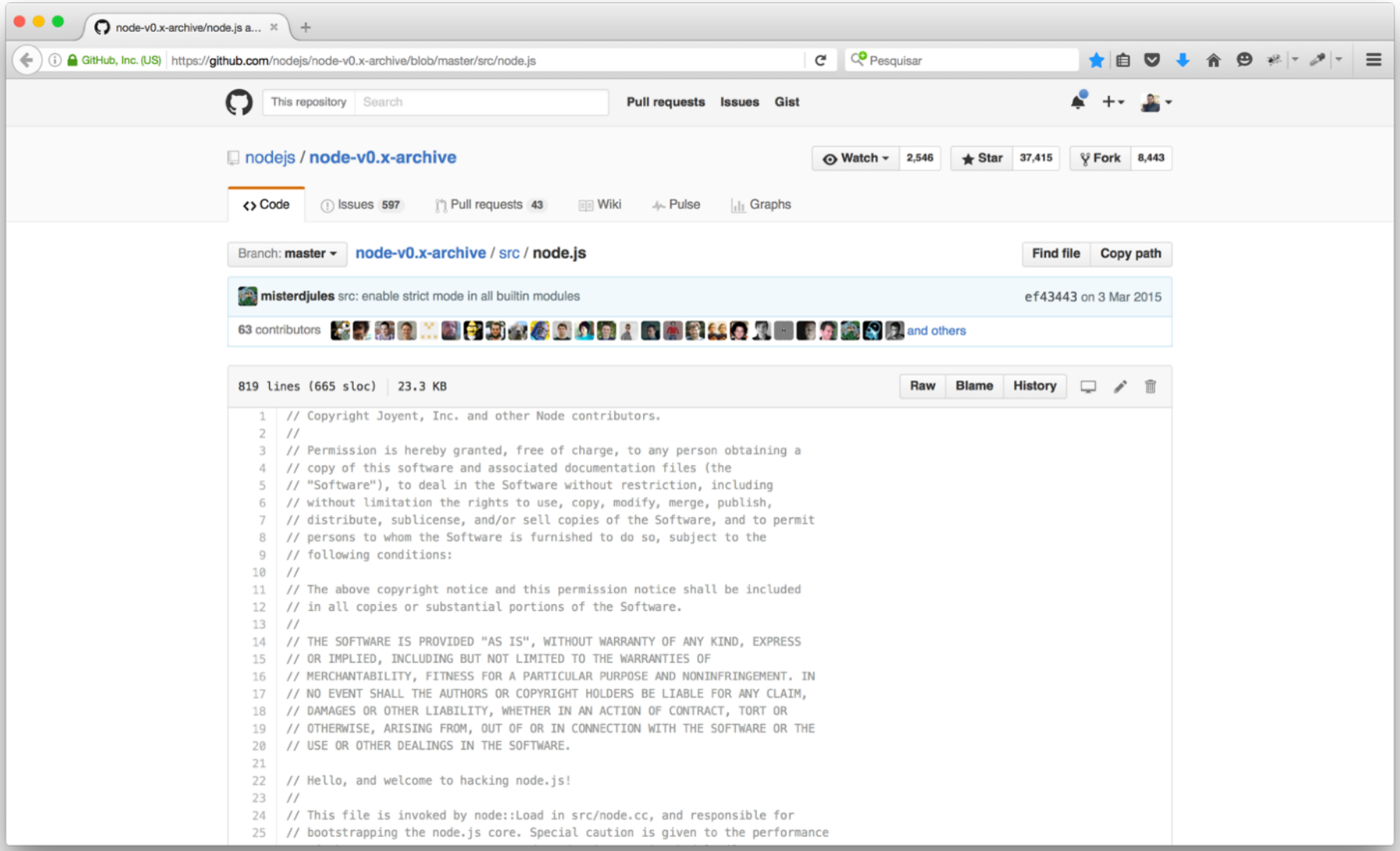

generator.js — nodejs
main.js
generator.js
1 const max = 100;
2
3 exports.generate = function () {
4 return Math.floor(Math.random() * max);
5 }
6
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
zsh
rodrigobranas nodejs $ node main.js
{ generate: [Function (anonymous)] }
rodrigobranas nodejs $


generator.js — nodejs
main.js
generator.js
1 const max = 100;
2
3 this.generate = function () {
4 return Math.floor(Math.random() * max);
5 }
6
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
zsh
rodrigobranas nodejs $ node main.js
{ generate: [Function (anonymous)] }
rodrigobranas nodejs $


Qual é a diferença entre
module.exports, exports e this?


main.js — nodejs
main.js
generator.js
1 console.log(module.exports === exports);
2 console.log(exports === this);
3 console.log(module.exports === this);
4
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
zsh
rodrigobranas nodejs $ node main.js
true
true
true
rodrigobranas nodejs $

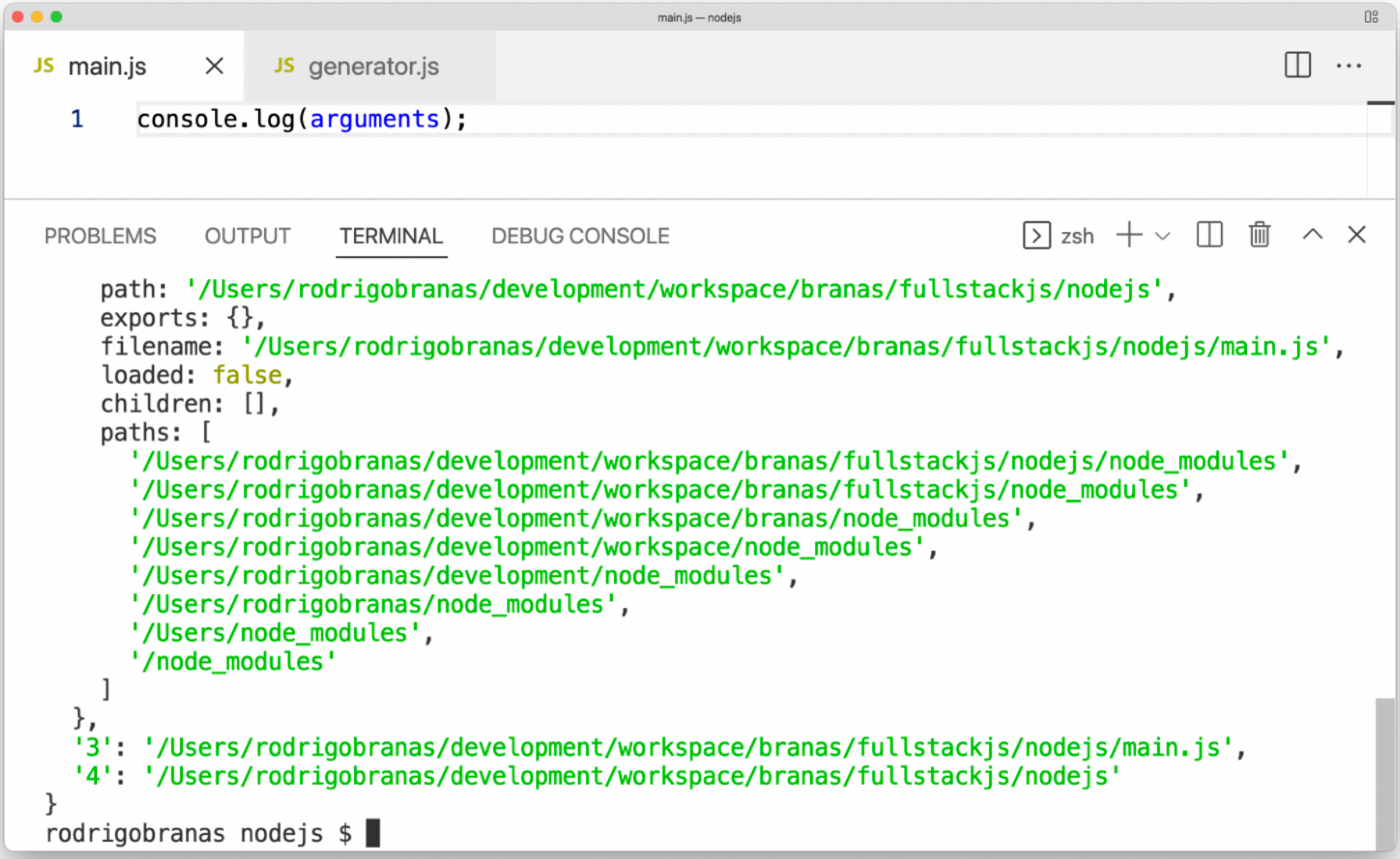
Cuidado, apenas module.exports é retornado da função require



https://github.com/nodejs/node-v0.x-archive/blob/master/src/node.js

```
748 function NativeModule(id) {
749   this.filename = id + '.js';
750   this.id = id;
751   this.exports = {};
752   this.loaded = false;
753 }
754
755 NativeModule._source = process.binding('natives');
756 NativeModule._cache = {};
757
758 NativeModule.require = function(id) {
759   if (id == 'native_module') {
760     return NativeModule;
761   }
762
763   var cached = NativeModule.getCached(id);
764   if (cached) {
765     return cached.exports;
766   }
767
768   if (!NativeModule.exists(id)) {
769     throw new Error('No such native module ' + id);
770   }
771
772   process.moduleLoadList.push('NativeModule ' + id);
773
774   var nativeModule = new NativeModule(id);
775
776   nativeModule.cache();
777   nativeModule.compile();
778
779   return nativeModule.exports;
780 };
```

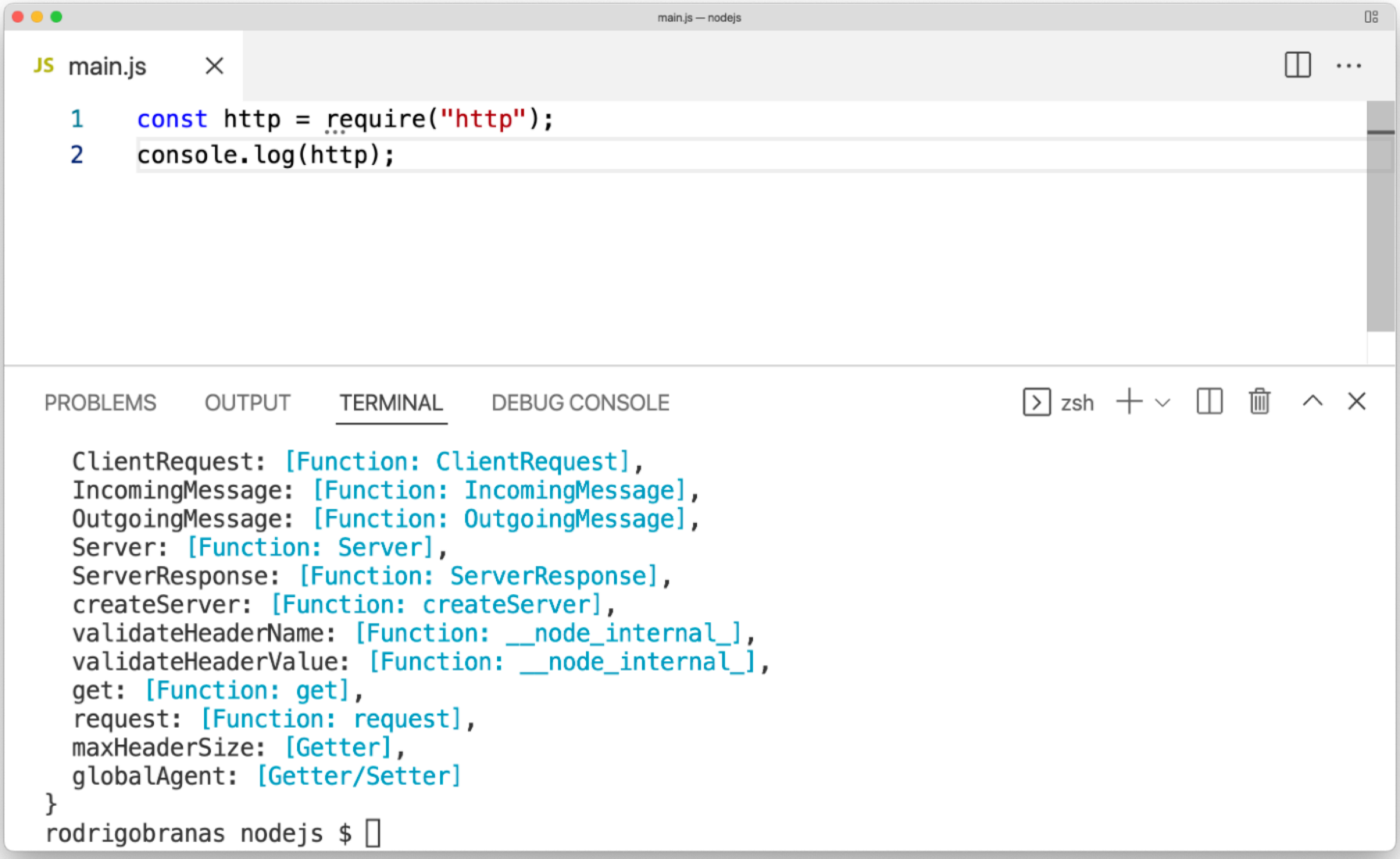
```
794  NativeModule.wrap = function(script) {
795    return NativeModule.wrapper[0] + script + NativeModule.wrapper[1];
796  };
797
798  NativeModule.wrapper = [
799    '(function (exports, require, module, __filename, __dirname) { ',
800    '\n});'
801  ];
802
803  NativeModule.prototype.compile = function() {
804    var source = NativeModule.getSource(this.id);
805    source = NativeModule.wrap(source);
806
807    var fn = runInThisContext(source, { filename: this.filename });
808    fn(this.exports, NativeModule.require, this, this.filename);
809
810    this.loaded = true;
811  };
812
```


main.js — nodejs
main.js
generator.js
1 console.log(arguments);
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
zsh
path: '/Users/rodrigobranas/development/workspace/branas/fullstackjs/nodejs',
exports: {},
filename: '/Users/rodrigobranas/development/workspace/branas/fullstackjs/nodejs/main.js',
loaded: false,
children: [],
paths: [
'/Users/rodrigobranas/development/workspace/branas/fullstackjs/nodejs/node_modules',
'/Users/rodrigobranas/development/workspace/branas/fullstackjs/node_modules',
'/Users/rodrigobranas/development/workspace/branas/node_modules',
'/Users/rodrigobranas/development/workspace/node_modules',
'/Users/rodrigobranas/development/node_modules',
'/Users/rodrigobranas/node_modules',
'/Users/node_modules',
'/node_modules'
]
},
'3': '/Users/rodrigobranas/development/workspace/branas/fullstackjs/nodejs/main.js',
'4': '/Users/rodrigobranas/development/workspace/branas/fullstackjs/nodejs'
}
rodrigobranas nodejs $

Como um módulo é **localizado**?

Primeiro, o algoritmo de busca tenta localizar um módulo core. Existem vários como: net, http, url, fs, zlib, crypto, events, stream, os, vm, util, entre outros.

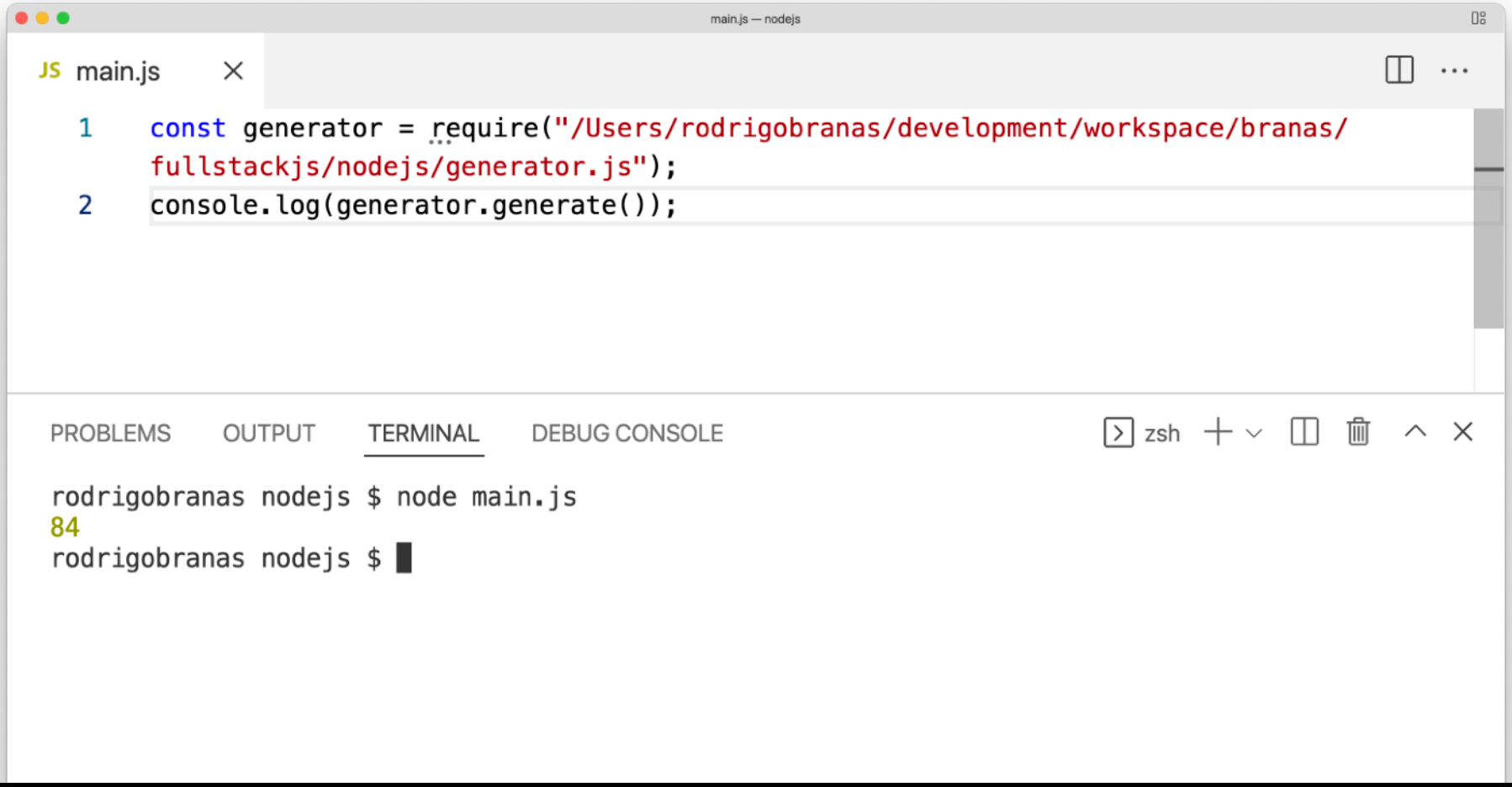
```
require(X) from module at path Y
1. If X is a core module,
   a. return the core module
   b. STOP
2. If X begins with './' or '/' or '../'
   a. LOAD_AS_FILE(Y + X)
   b. LOAD_AS_DIRECTORY(Y + X)
3. LOAD_NODE_MODULES(X, dirname(Y))
4. THROW "not found"

LOAD_AS_FILE(X)
1. If X is a file, load X as JavaScript text.  STOP
2. If X.js is a file, load X.js as JavaScript text.  STOP
3. If X.json is a file, parse X.json to a JavaScript Object.  STOP
4. If X.node is a file, load X.node as binary addon.  STOP

LOAD_AS_DIRECTORY(X)
1. If X/package.json is a file,
   a. Parse X/package.json, and look for "main" field.
   b. let M = X + (json main field)
   c. LOAD_AS_FILE(M)
2. If X/index.js is a file, load X/index.js as JavaScript text.  STOP
3. If X/index.json is a file, parse X/index.json to a JavaScript object. STOP
4. If X/index.node is a file, load X/index.node as binary addon.  STOP
```


main.js — nodejs
main.js
1 const http = require("http");
2 console.log(http);
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
zsh
ClientRequest: [Function: ClientRequest],
IncomingMessage: [Function: IncomingMessage],
OutgoingMessage: [Function: OutgoingMessage],
Server: [Function: Server],
ServerResponse: [Function: ServerResponse],
createServer: [Function: createServer],
validateHeaderName: [Function: __node_internal_],
validateHeaderValue: [Function: __node_internal_],
get: [Function: get],
request: [Function: request],
maxHeaderSize: [Getter],
globalAgent: [Getter/Setter]
}
rodrigobranas nodejs $

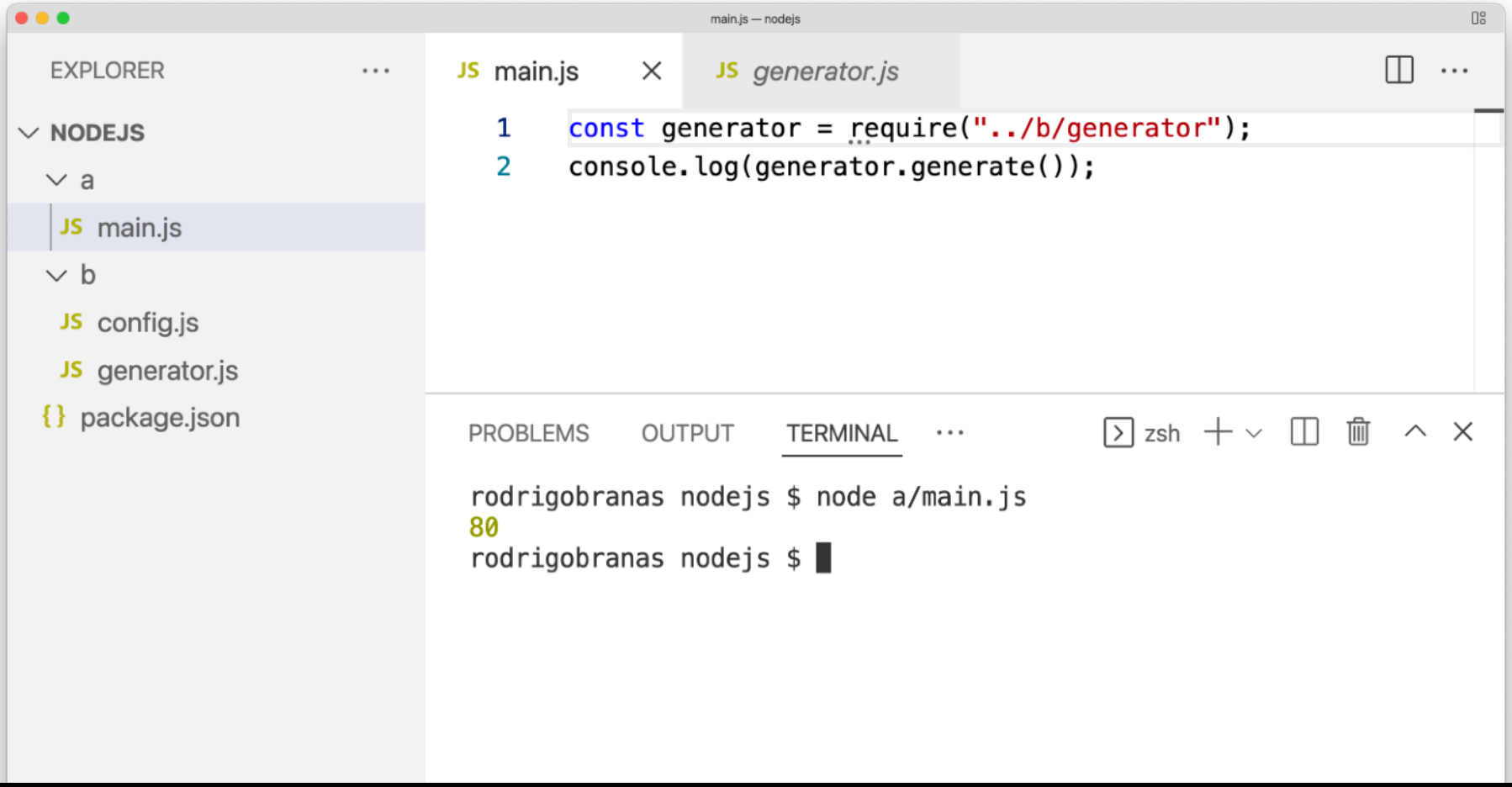
Se o nome do módulo iniciar com '/', '../' ou './', o algoritmo de busca irá localizar o módulo por meio do caminho absoluto no sistema de arquivos.

```
require(X) from module at path Y
1. If X is a core module,
   a. return the core module
   b. STOP
2. If X begins with './' or '/' or '../'
   a. LOAD_AS_FILE(Y + X)
   b. LOAD_AS_DIRECTORY(Y + X)
3. LOAD_NODE_MODULES(X, dirname(Y))
4. THROW "not found"

LOAD_AS_FILE(X)
1. If X is a file, load X as JavaScript text.  STOP
2. If X.js is a file, load X.js as JavaScript text.  STOP
3. If X.json is a file, parse X.json to a JavaScript Object.  STOP
4. If X.node is a file, load X.node as binary addon.  STOP

LOAD_AS_DIRECTORY(X)
1. If X/package.json is a file,
   a. Parse X/package.json, and look for "main" field.
   b. let M = X + (json main field)
   c. LOAD_AS_FILE(M)
2. If X/index.js is a file, load X/index.js as JavaScript text.  STOP
3. If X/index.json is a file, parse X/index.json to a JavaScript object. STOP
4. If X/index.node is a file, load X/index.node as binary addon.  STOP
```

Localizando um módulo na mesma pasta



Localizando um módulo por meio do caminho absoluto



Localizando um módulo por meio do caminho relativo

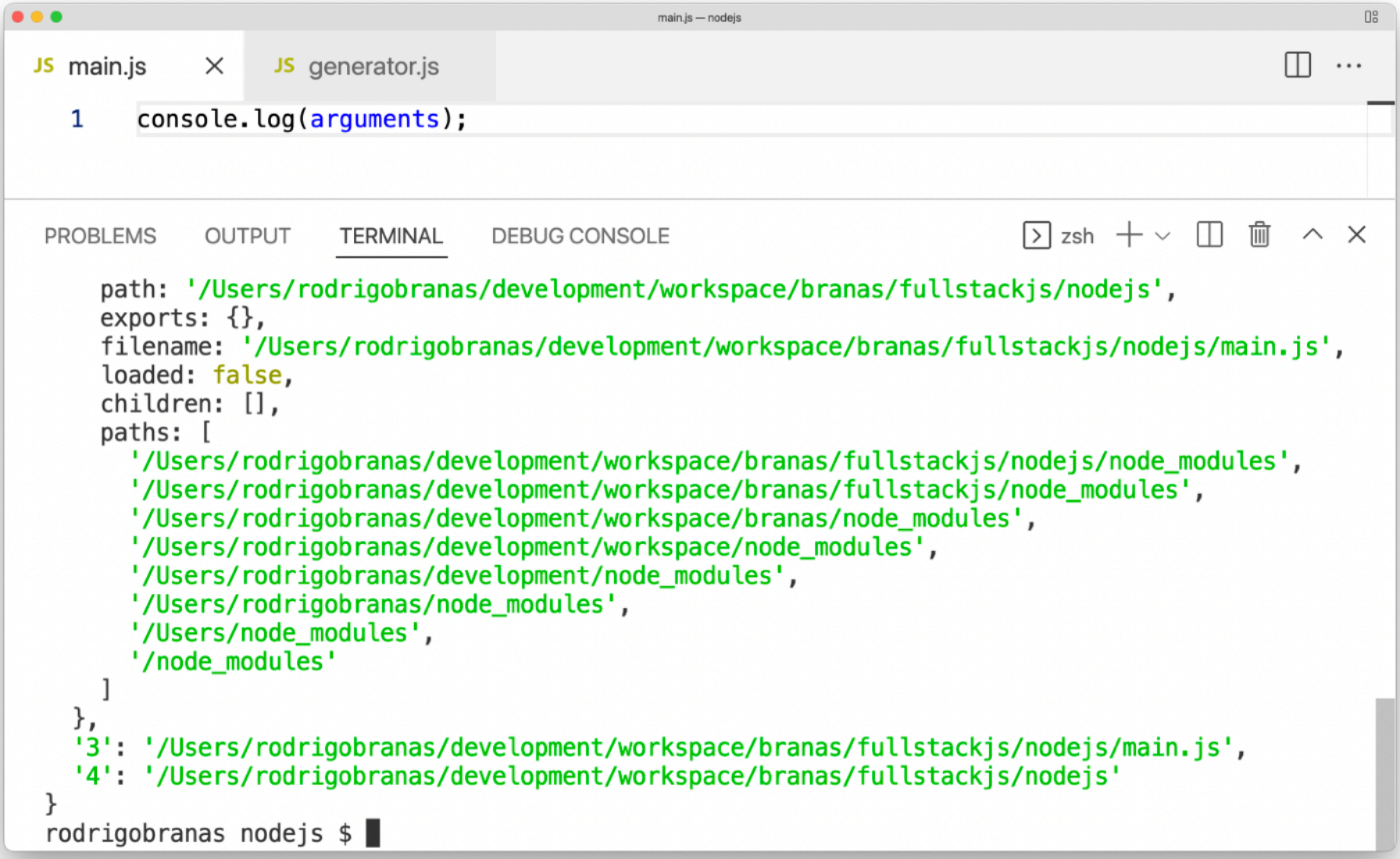
Terminar o nome com '.js' é opcional, o algoritmo de busca irá tentar colocar o '.js' no final caso não encontre o módulo.

```
require(X) from module at path Y
1. If X is a core module,
   a. return the core module
   b. STOP
2. If X begins with './' or '/' or '../'
   a. LOAD_AS_FILE(Y + X)
   b. LOAD_AS_DIRECTORY(Y + X)
3. LOAD_NODE_MODULES(X, dirname(Y))
4. THROW "not found"

LOAD_AS_FILE(X)
1. If X is a file, load X as JavaScript text.  STOP
2. If X.js is a file, load X.js as JavaScript text.  STOP
3. If X.json is a file, parse X.json to a JavaScript Object.  STOP
4. If X.node is a file, load X.node as binary addon.  STOP

LOAD_AS_DIRECTORY(X)
1. If X/package.json is a file,
   a. Parse X/package.json, and look for "main" field.
   b. let M = X + (json main field)
   c. LOAD_AS_FILE(M)
2. If X/index.js is a file, load X/index.js as JavaScript text.  STOP
3. If X/index.json is a file, parse X/index.json to a JavaScript object. STOP
4. If X/index.node is a file, load X/index.node as binary addon.  STOP
```

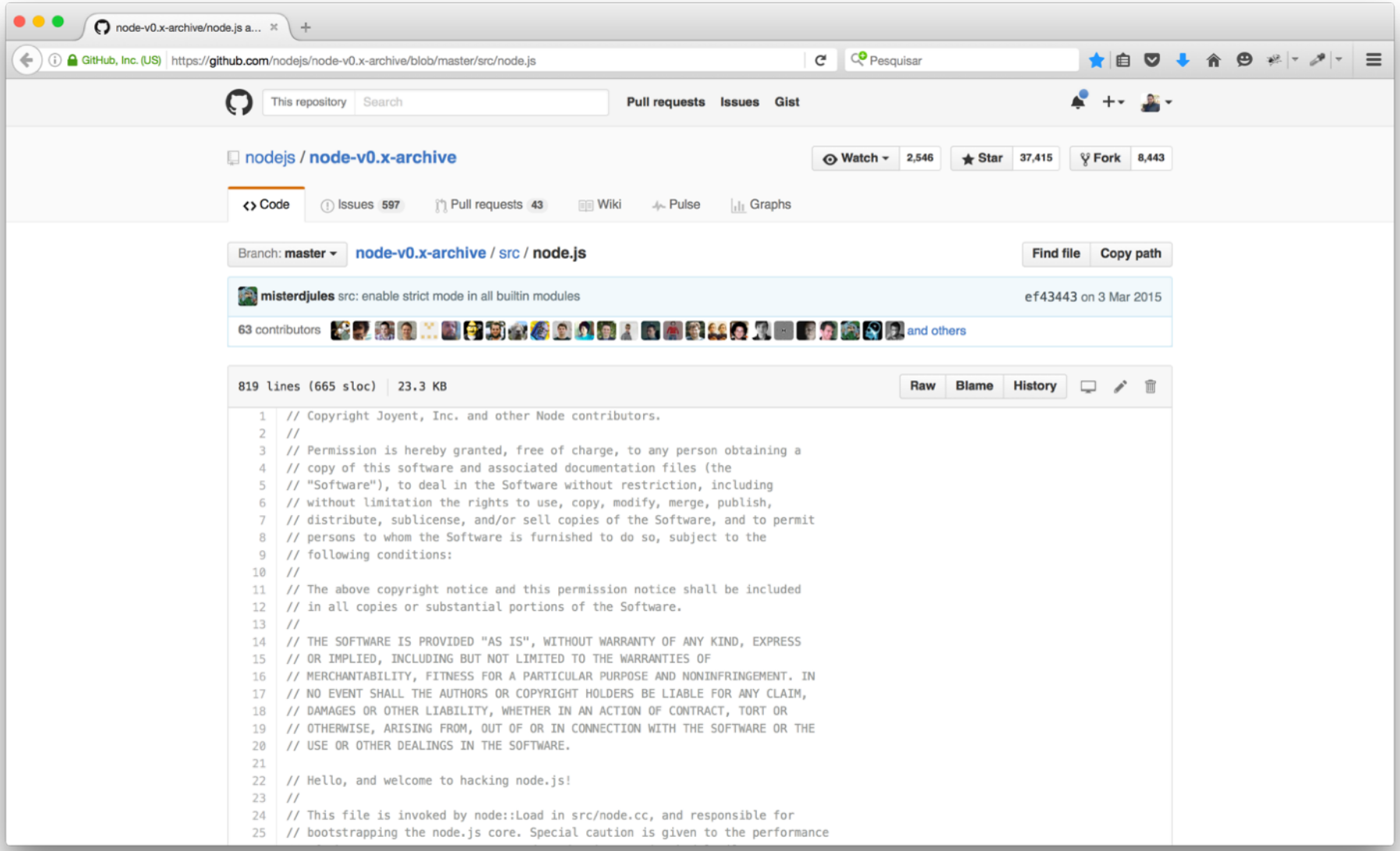
Se o módulo não começar com '/', '../' ou './' o algoritmo de busca da função require vai procurar dentro da pasta node_modules.

```
require(X) from module at path Y
1. If X is a core module,
   a. return the core module
   b. STOP
2. If X begins with './' or '/' or '../'
   a. LOAD_AS_FILE(Y + X)
   b. LOAD_AS_DIRECTORY(Y + X)
3. LOAD_NODE_MODULES(X, dirname(Y))
4. THROW "not found"

LOAD_AS_FILE(X)
1. If X is a file, load X as JavaScript text.  STOP
2. If X.js is a file, load X.js as JavaScript text.  STOP
3. If X.json is a file, parse X.json to a JavaScript Object.  STOP
4. If X.node is a file, load X.node as binary addon.  STOP

LOAD_AS_DIRECTORY(X)
1. If X/package.json is a file,
   a. Parse X/package.json, and look for "main" field.
   b. let M = X + (json main field)
   c. LOAD_AS_FILE(M)
2. If X/index.js is a file, load X/index.js as JavaScript text.  STOP
3. If X/index.json is a file, parse X/index.json to a JavaScript object. STOP
4. If X/index.node is a file, load X/index.node as binary addon.  STOP
```

```
LOAD_NODE_MODULES(X, START)
1. let DIRS=NODE_MODULES_PATHS(START)
2. for each DIR in DIRS:
   a. LOAD_AS_FILE(DIR/X)
   b. LOAD_AS_DIRECTORY(DIR/X)

NODE_MODULES_PATHS(START)
1. let PARTS = path split(START)
2. let I = count of PARTS - 1
3. let DIRS = []
4. while I >= 0,
   a. if PARTS[I] = "node_modules" CONTINUE
   c. DIR = path join(PARTS[0 .. I] + "node_modules")
   b. DIRS = DIRS + DIR
   c. let I = I - 1
5. return DIRS
```

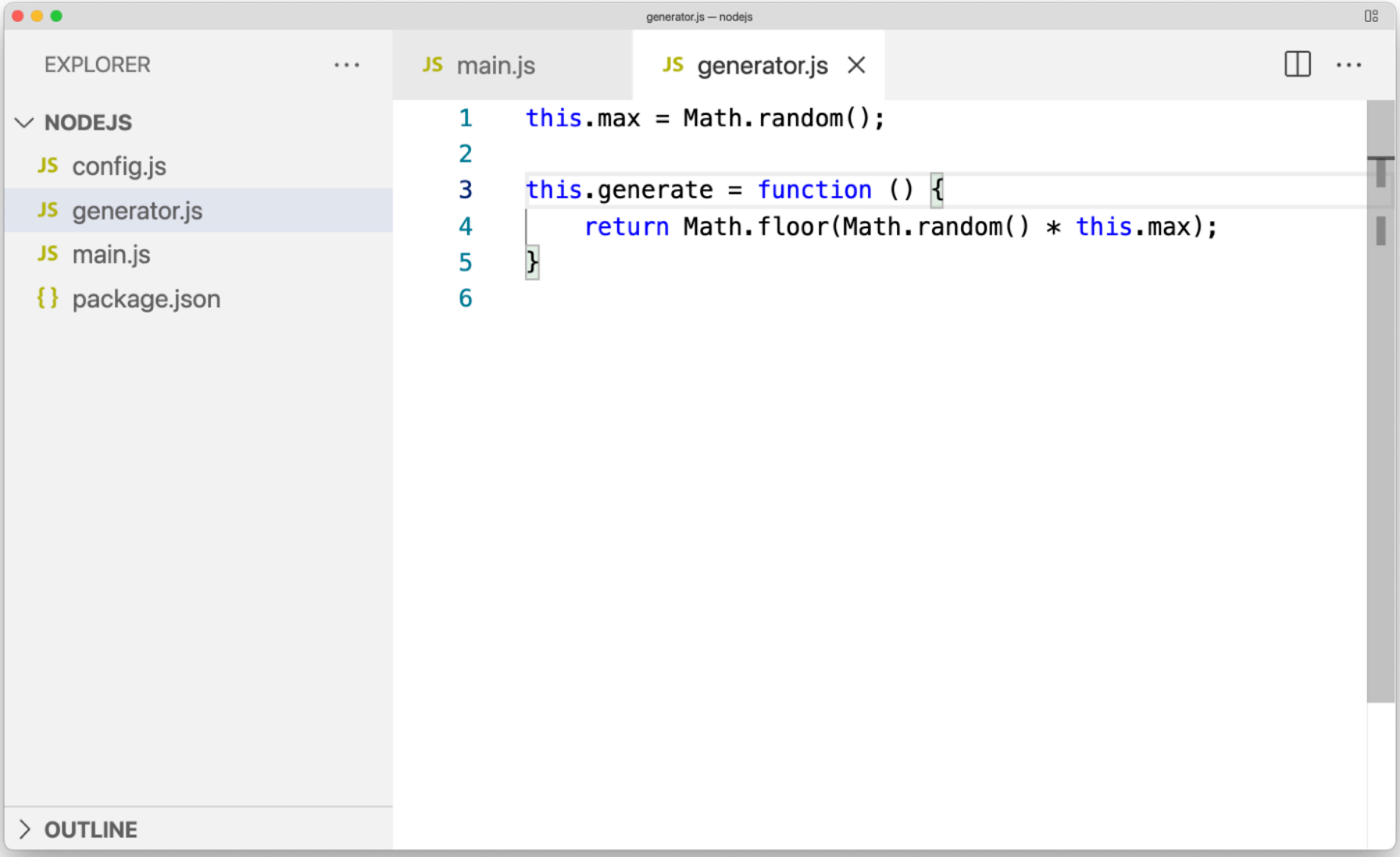

main.js — nodejs
main.js
generator.js
1 console.log(arguments);
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
zsh
path: '/Users/rodrigobranas/development/workspace/branas/fullstackjs/nodejs',
exports: {},
filename: '/Users/rodrigobranas/development/workspace/branas/fullstackjs/nodejs/main.js',
loaded: false,
children: [],
paths: [
'/Users/rodrigobranas/development/workspace/branas/fullstackjs/nodejs/node_modules',
'/Users/rodrigobranas/development/workspace/branas/fullstackjs/node_modules',
'/Users/rodrigobranas/development/workspace/branas/node_modules',
'/Users/rodrigobranas/development/workspace/node_modules',
'/Users/rodrigobranas/development/node_modules',
'/Users/rodrigobranas/node_modules',
'/Users/node_modules',
'/node_modules'
]
},
'3': '/Users/rodrigobranas/development/workspace/branas/fullstackjs/nodejs/main.js',
'4': '/Users/rodrigobranas/development/workspace/branas/fullstackjs/nodejs'
}
rodrigobranas nodejs $

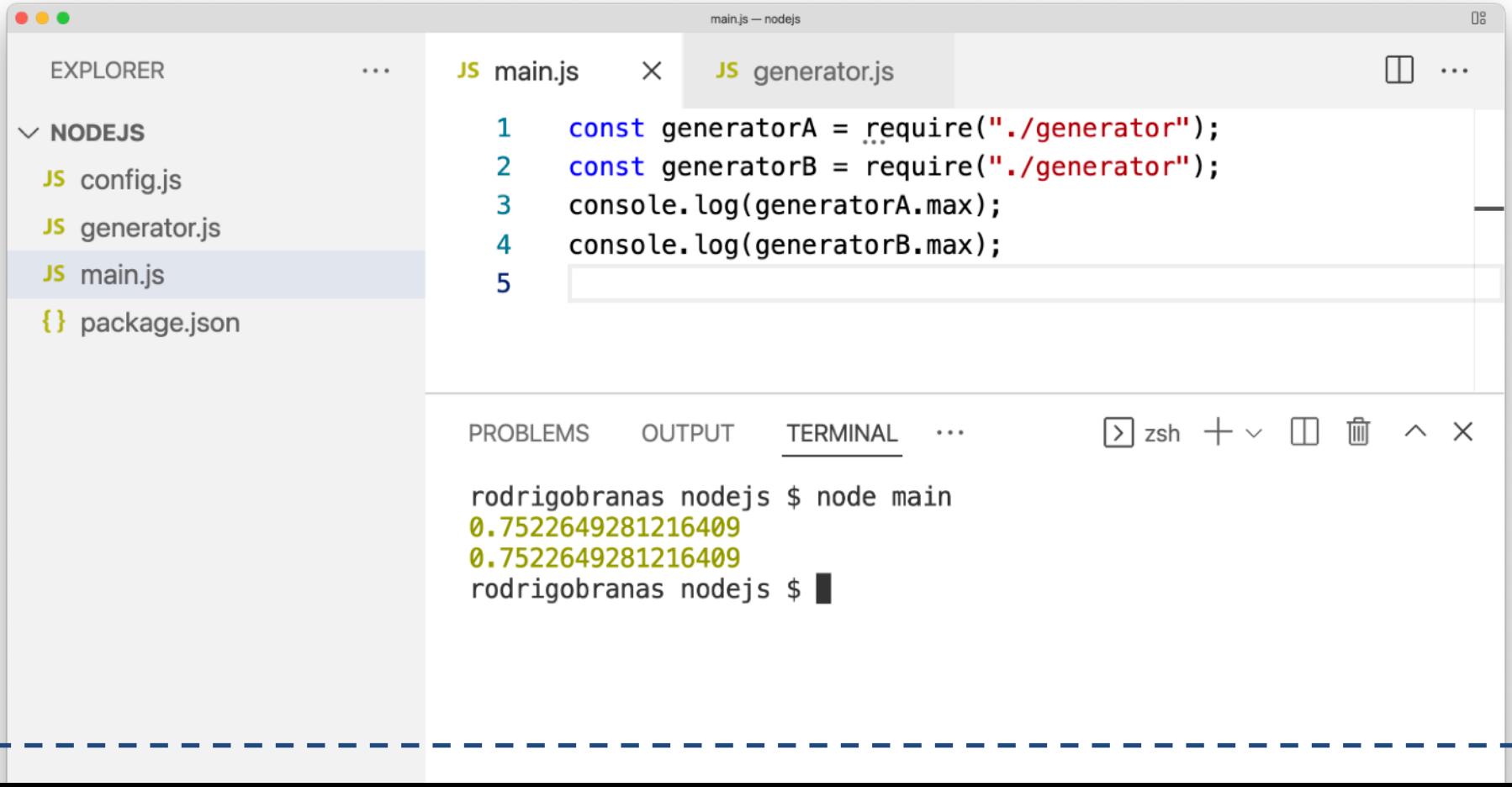
A função require utiliza um mecanismo de cache



https://github.com/nodejs/node-v0.x-archive/blob/master/src/node.js

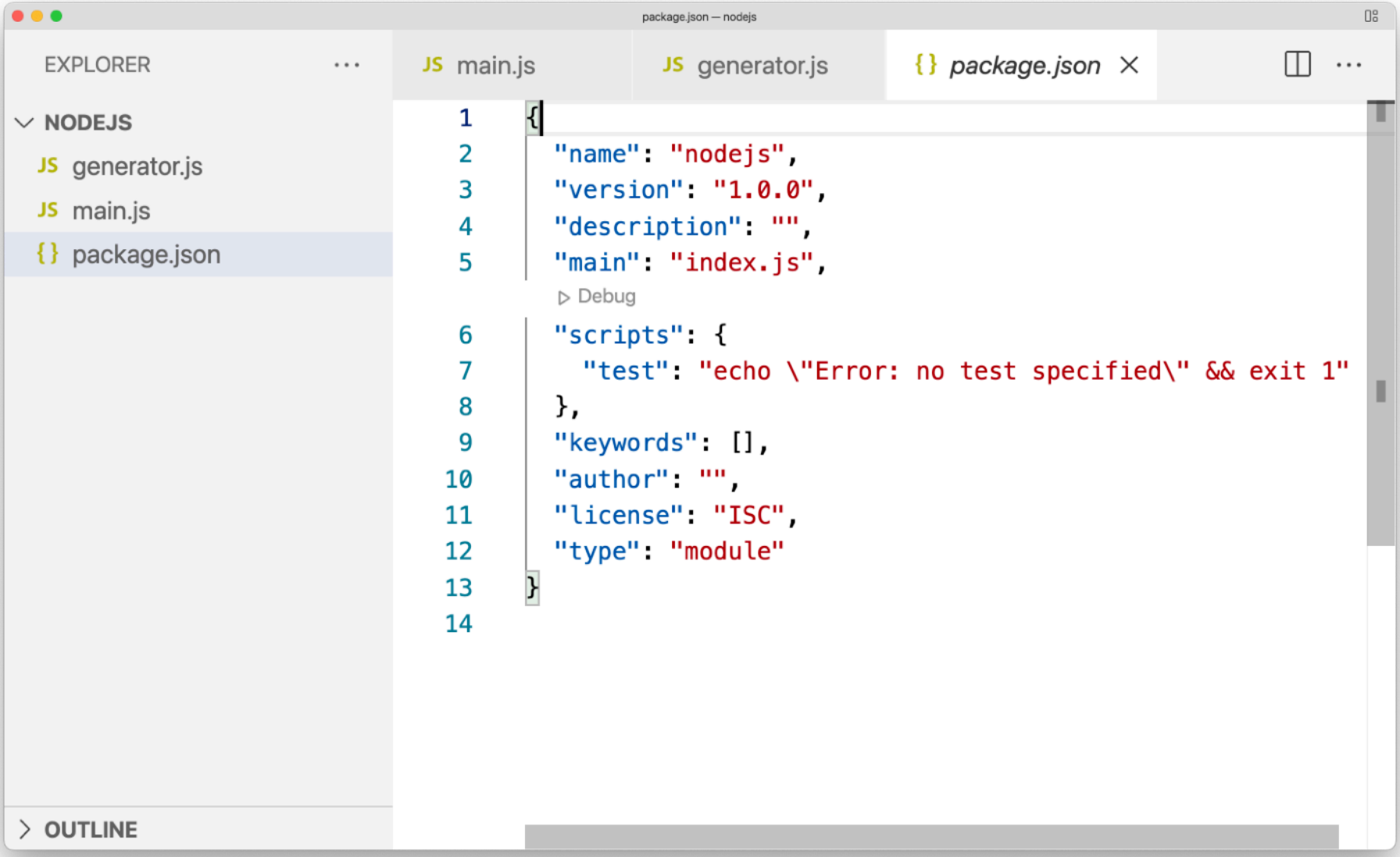
```
758   NativeModule.require = function(id) {
759     if (id == 'native_module') {
760       return NativeModule;
761     }
762
763     var cached = NativeModule.getCached(id);
764     if (cached) {
765       return cached.exports;
766     }
767
768     if (!NativeModule.exists(id)) {
769       throw new Error('No such native module ' + id);
770     }
771
772     process.moduleLoadList.push('NativeModule ' + id);
773
774     var nativeModule = new NativeModule(id);
775
776     nativeModule.cache();
777     nativeModule.compile();
778
779     return nativeModule.exports;
780   };
```

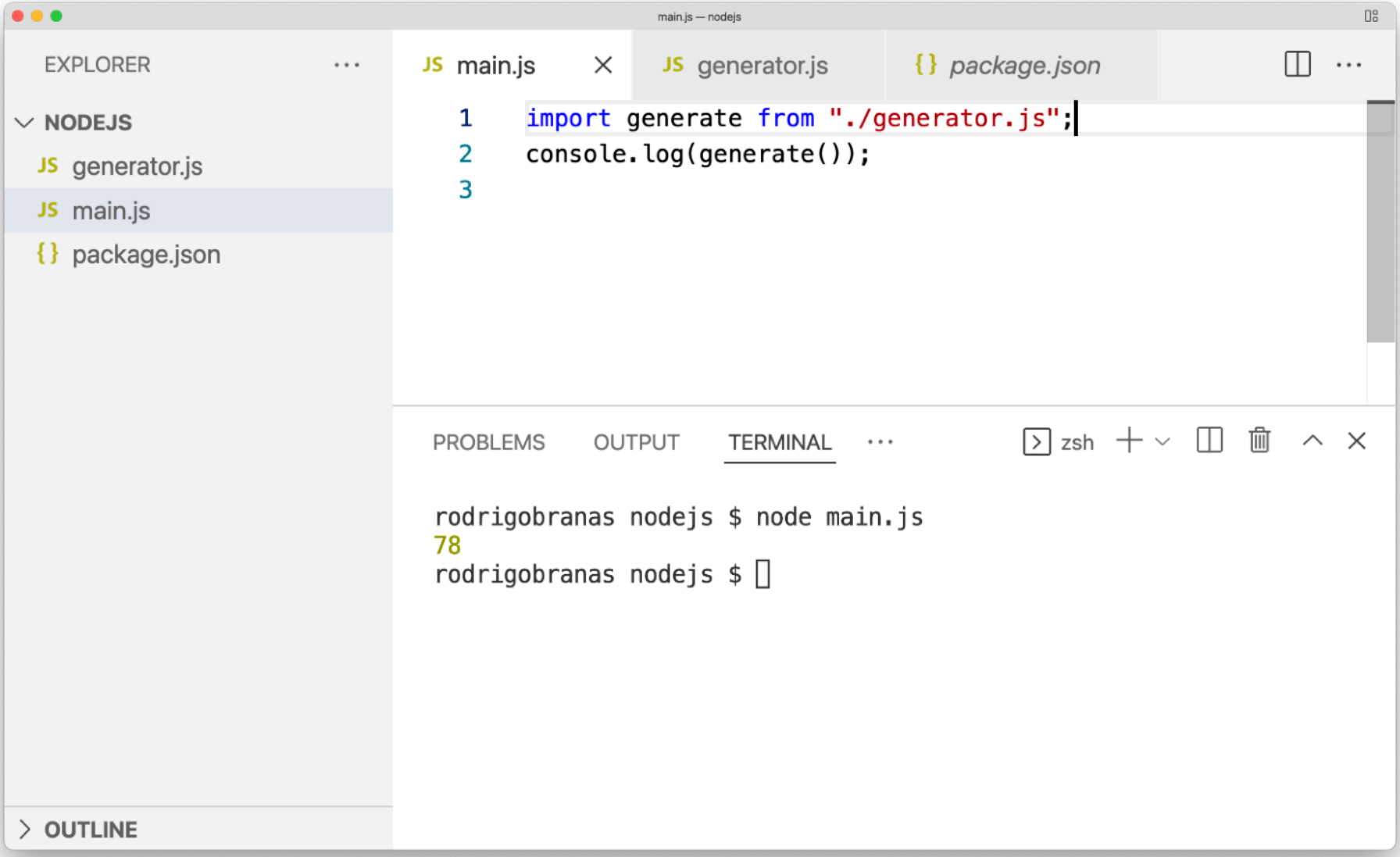

generator.js — nodejs
EXPLORER
NODEJS
config.js
generator.js
main.js
package.json
OUTLINE
main.js
generator.js
this.max = Math.random();
this.generate = function () {
return Math.floor(Math.random() * this.max);
}

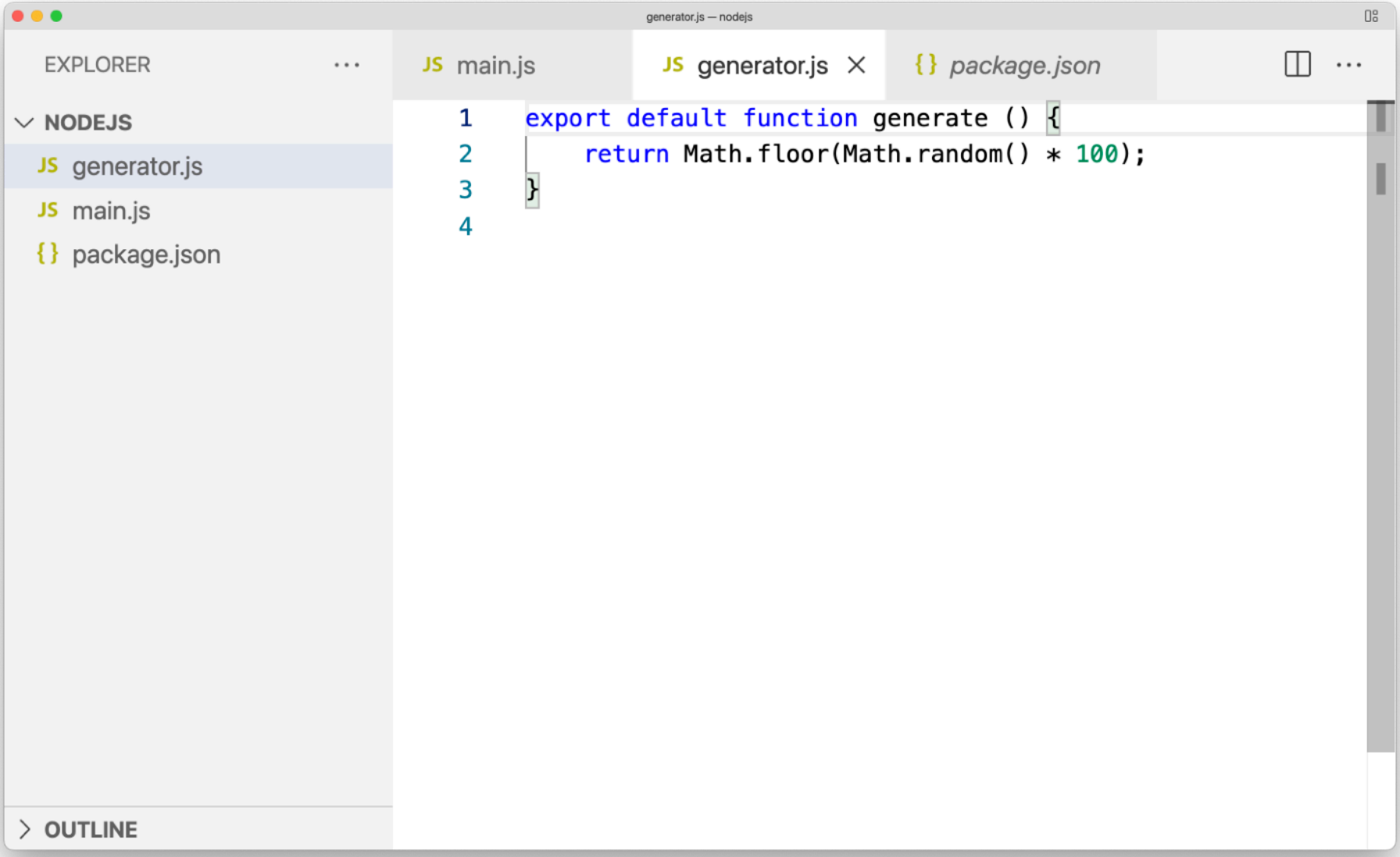


O módulo serialGenerator ficou armazenado em cache

Há alguns anos o Node.js dá suporte para os ES Modules, começando com a extensão .mjs e agora permitindo a configuração no package.json por meio da propriedade type e o valor module

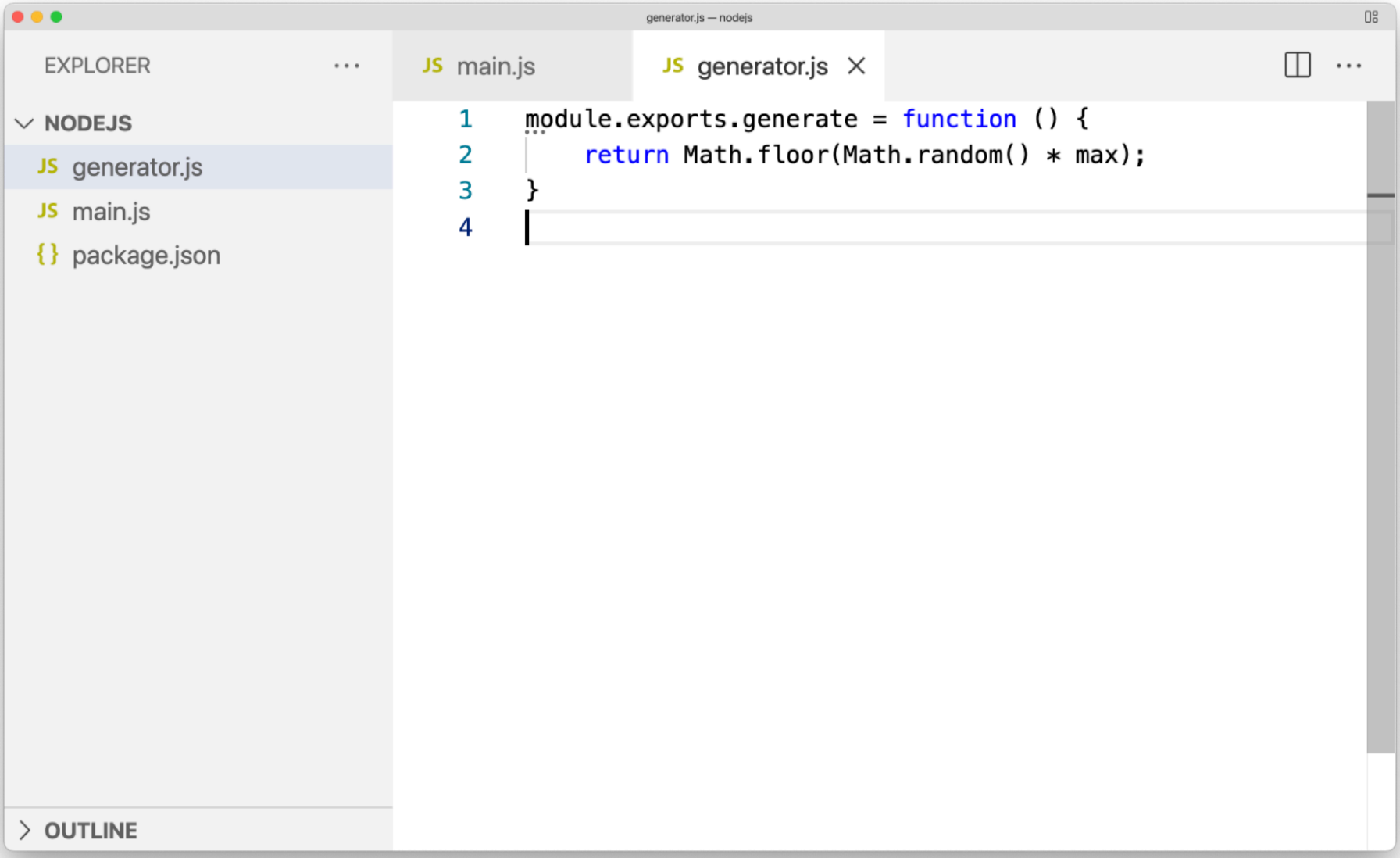

package.json — nodejs
EXPLORER
NODEJS
generator.js
main.js
package.json
OUTLINE
main.js
generator.js
package.json
{
"name": "nodejs",
"version": "1.0.0",
"description": "",
"main": "index.js",
Debug
"scripts": {
"test": "echo \"Error: no test specified\" && exit 1"
},
"keywords": [],
"author": "",
"license": "ISC",
"type": "module"
}

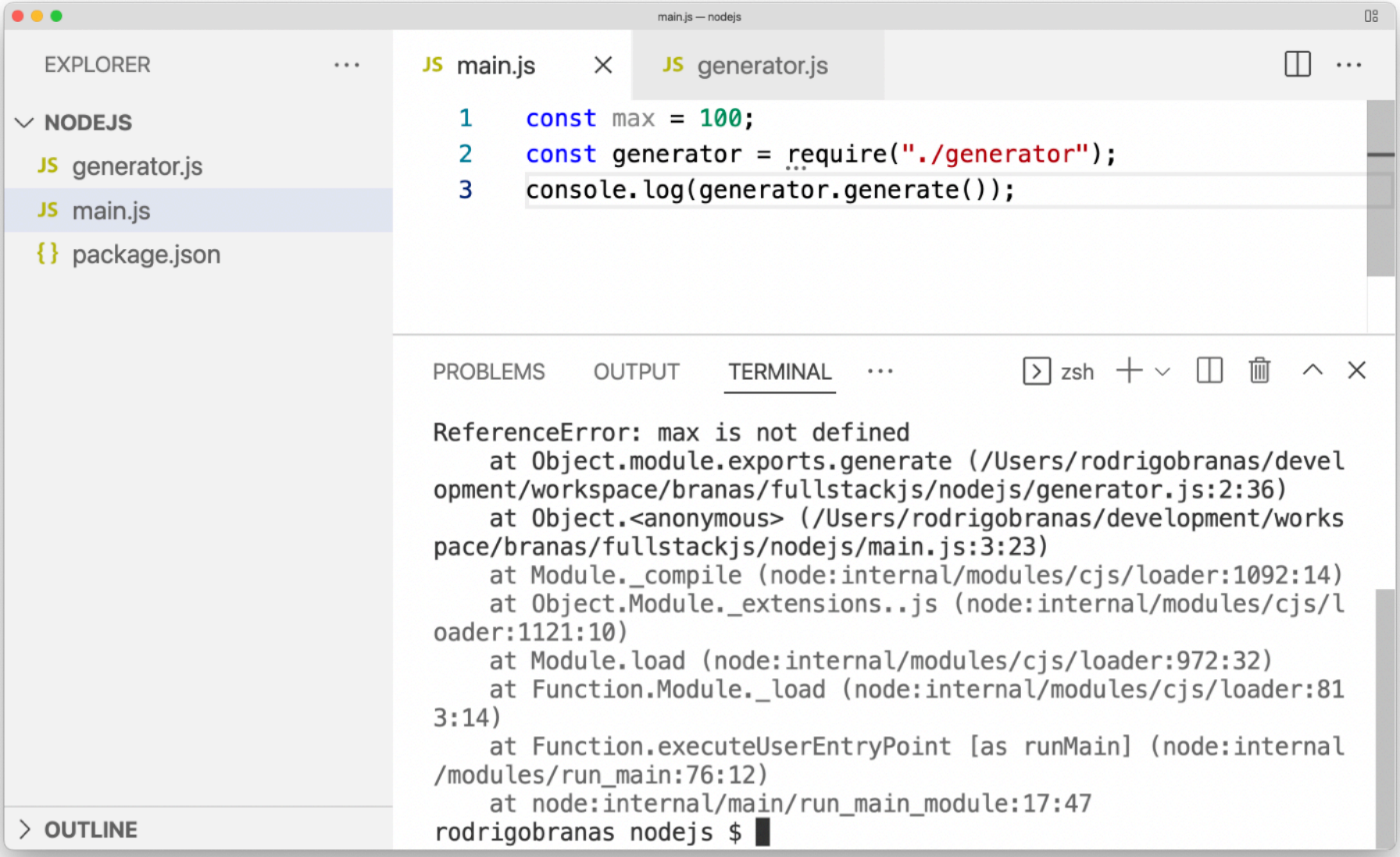

main.js — nodejs
EXPLORER
NODEJS
generator.js
main.js
package.json
OUTLINE
main.js
generator.js
package.json
1 import generate from "./generator.js";
2 console.log(generate());
3
PROBLEMS
OUTPUT
TERMINAL
zsh
rodrigobranas nodejs $ node main.js
78
rodrigobranas nodejs $


generator.js — nodejs
EXPLORER
NODEJS
generator.js
main.js
package.json
OUTLINE
main.js
generator.js
package.json
1 export default function generate () {
2 return Math.floor(Math.random() * 100);
3 }
4

Global Objects

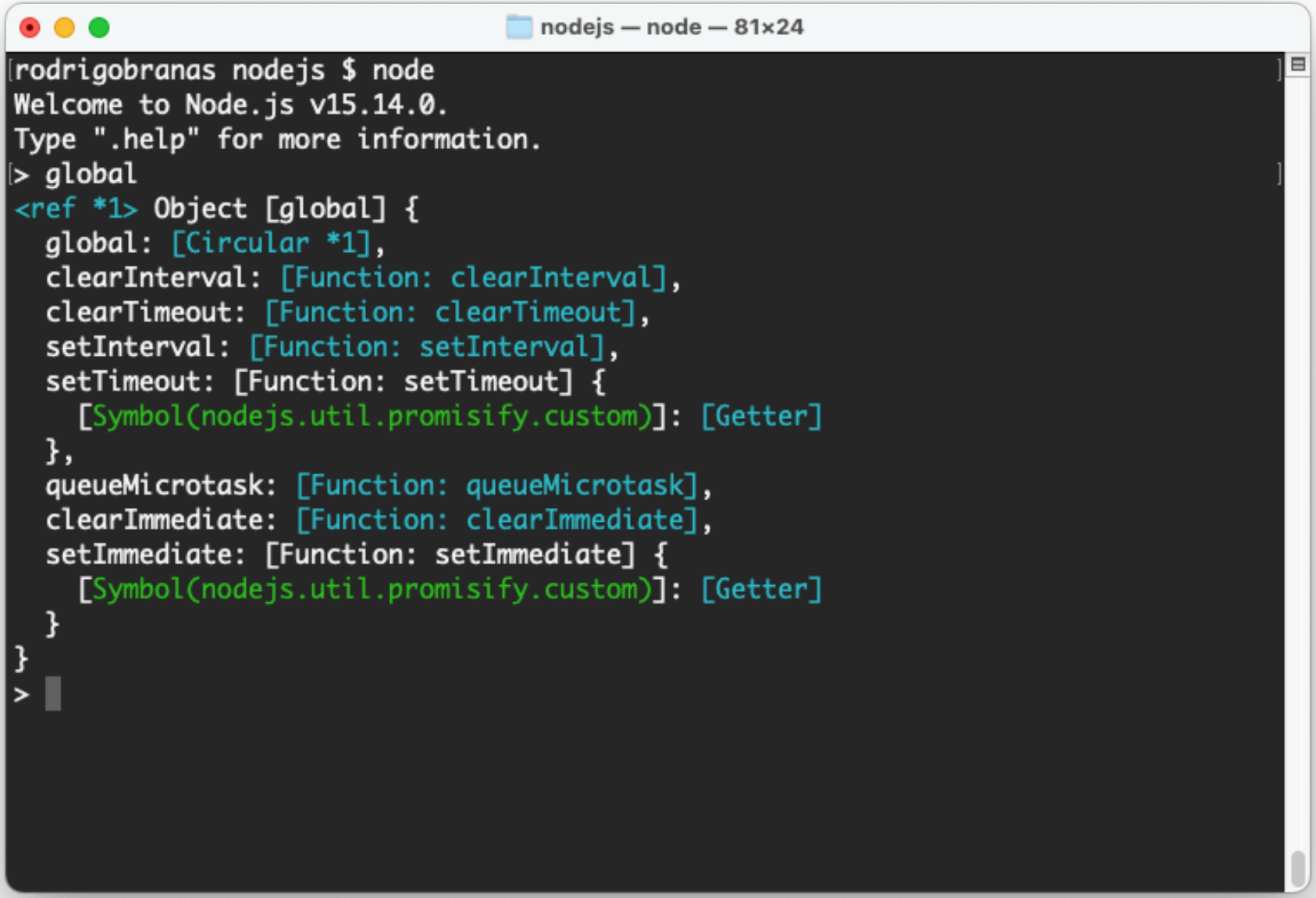
Tudo que é definido dentro do módulo por padrão é privado

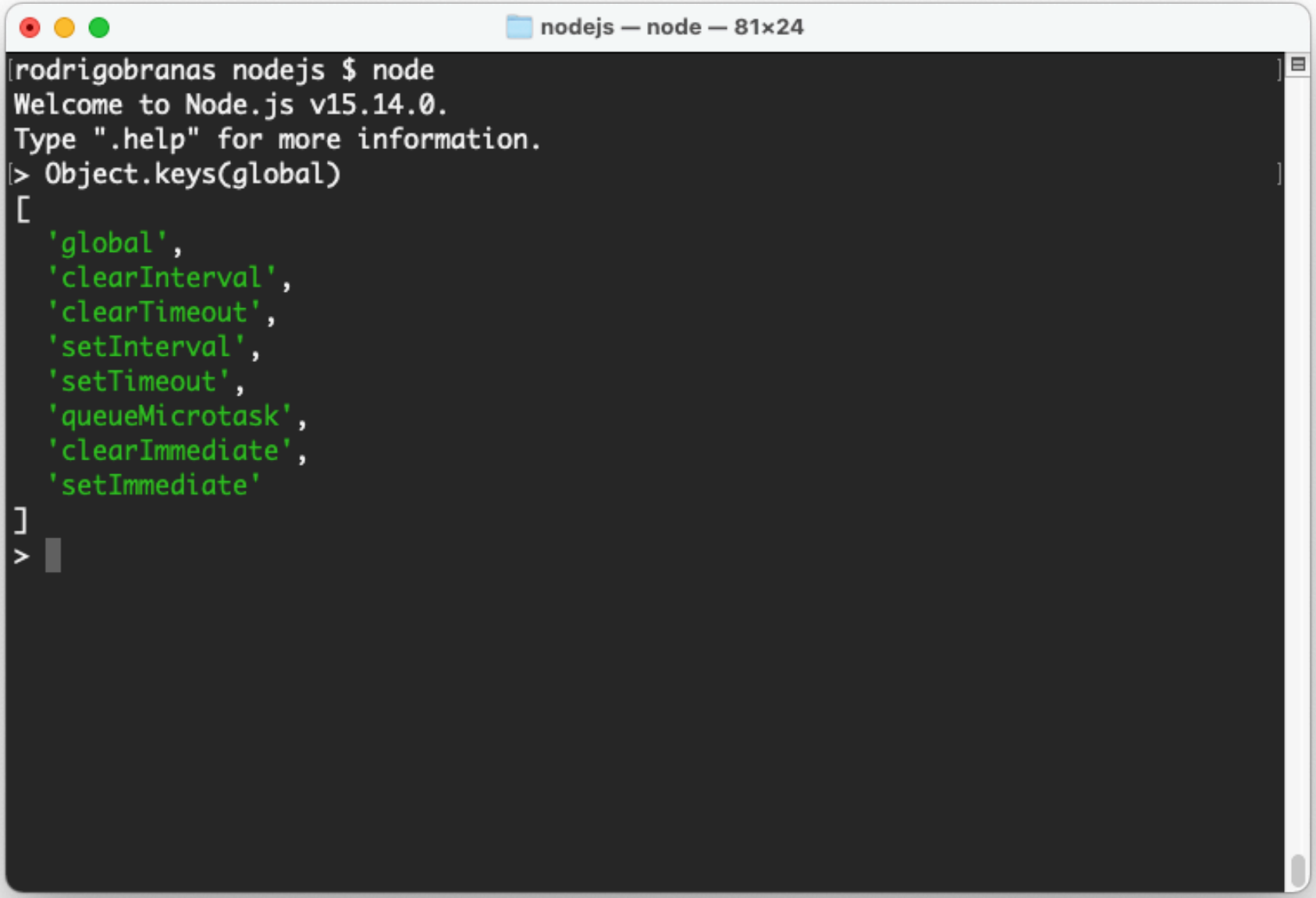

generator.js — nodejs
EXPLORER
NODEJS
generator.js
main.js
package.json
OUTLINE
main.js
generator.js
1 module.exports.generate = function () {
2 return Math.floor(Math.random() * max);
3 }
4


main.js — nodejs
EXPLORER
NODEJS
generator.js
main.js
package.json
OUTLINE
main.js
generator.js
1 const max = 100;
2 const generator = require("./generator");
3 console.log(generator.generate());
PROBLEMS
OUTPUT
TERMINAL
zsh
ReferenceError: max is not defined
at Object.module.exports.generate (/Users/rodrigobranas/development/workspace/branas/fullstackjs/nodejs/generator.js:2:36)
at Object.<anonymous> (/Users/rodrigobranas/development/workspace/branas/fullstackjs/nodejs/main.js:3:23)
at Module._compile (node:internal/modules/cjs/loader:1092:14)
at Object.Module._extensions..js (node:internal/modules/cjs/loader:1121:10)
at Module.load (node:internal/modules/cjs/loader:972:32)
at Function.Module._load (node:internal/modules/cjs/loader:813:14)
at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:76:12)
at node:internal/main/run_main_module:17:47
rodrigobranas nodejs $

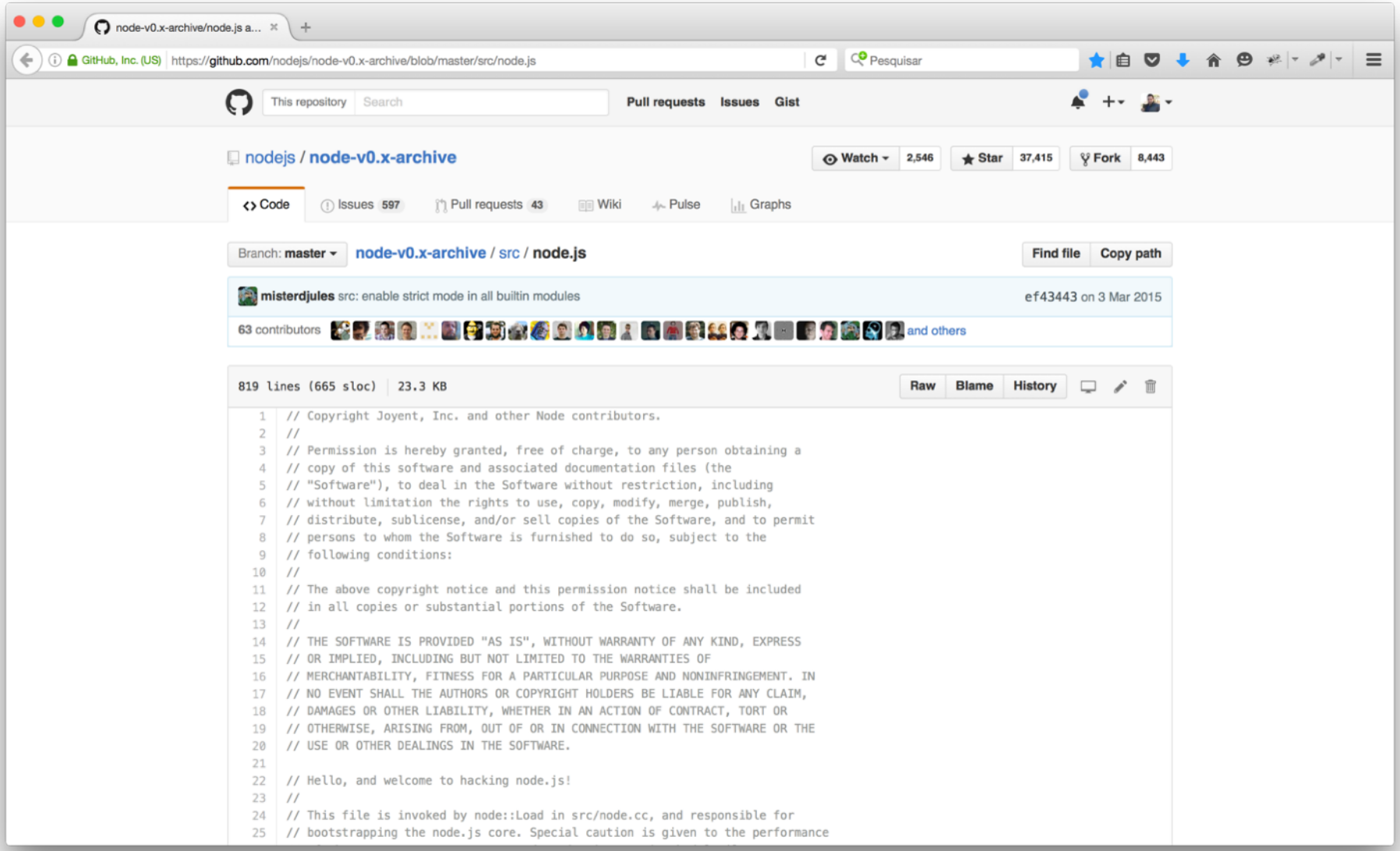
Existe um escopo global, similar ao dos navegadores?


Cuidado, evite poluir o escopo global


nodejs — node — 81×24
rodrigobranas nodejs $ node
Welcome to Node.js v15.14.0.
Type ".help" for more information.
> global
<ref *1> Object [global] {
global: [Circular *1],
clearInterval: [Function: clearInterval],
clearTimeout: [Function: clearTimeout],
setInterval: [Function: setInterval],
setTimeout: [Function: setTimeout] {
[Symbol(nodejs.util.promisify.custom)]: [Getter]
},
queueMicrotask: [Function: queueMicrotask],
clearImmediate: [Function: clearImmediate],
setImmediate: [Function: setImmediate] {
[Symbol(nodejs.util.promisify.custom)]: [Getter]
}
}
>

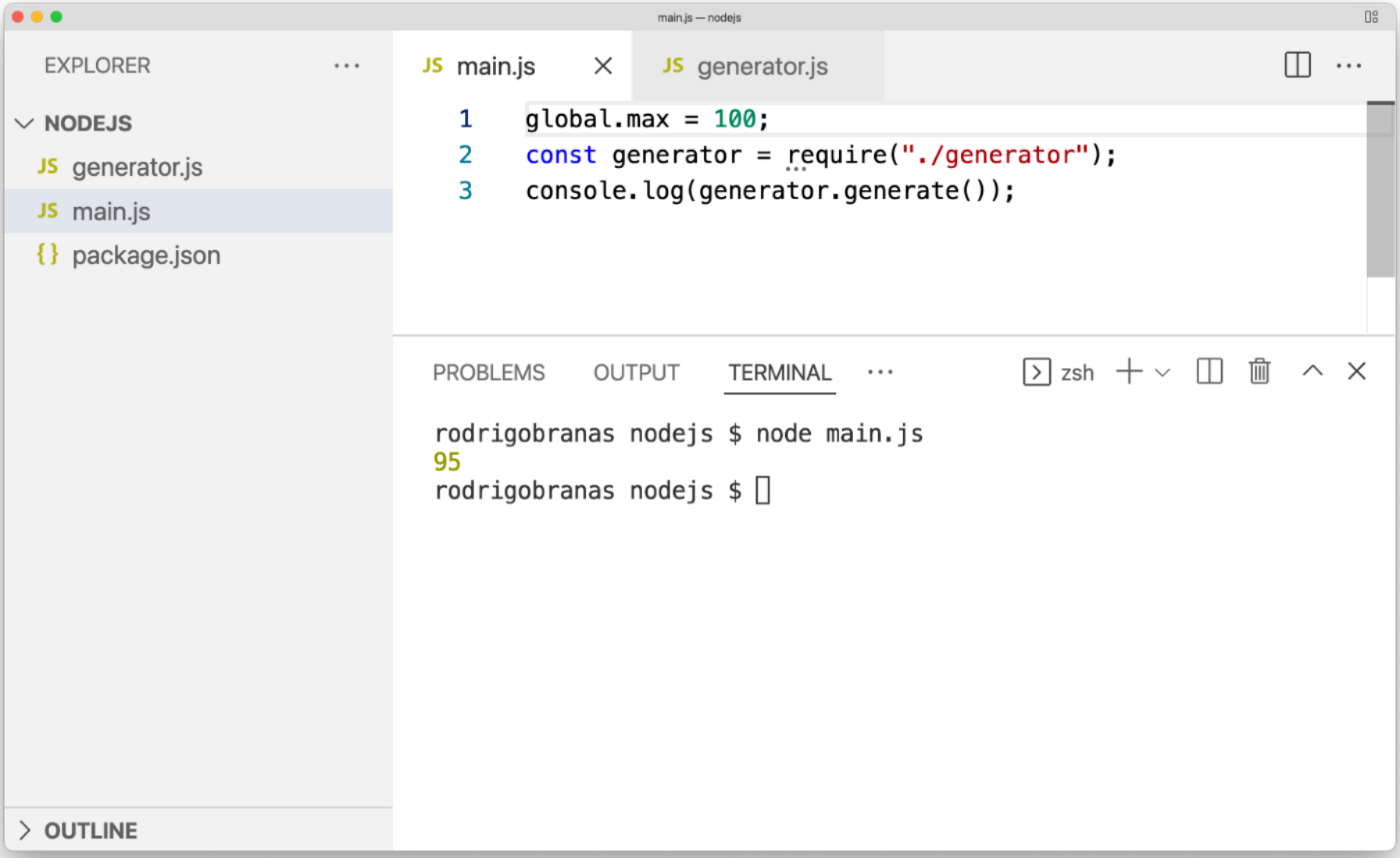

nodejs — node — 81×24
rodrigobranas nodejs $ node
Welcome to Node.js v15.14.0.
Type ".help" for more information.
> Object.keys(global)
[
'global',
'clearInterval',
'clearTimeout',
'setInterval',
'setTimeout',
'queueMicrotask',
'clearImmediate',
'setImmediate'
]
>

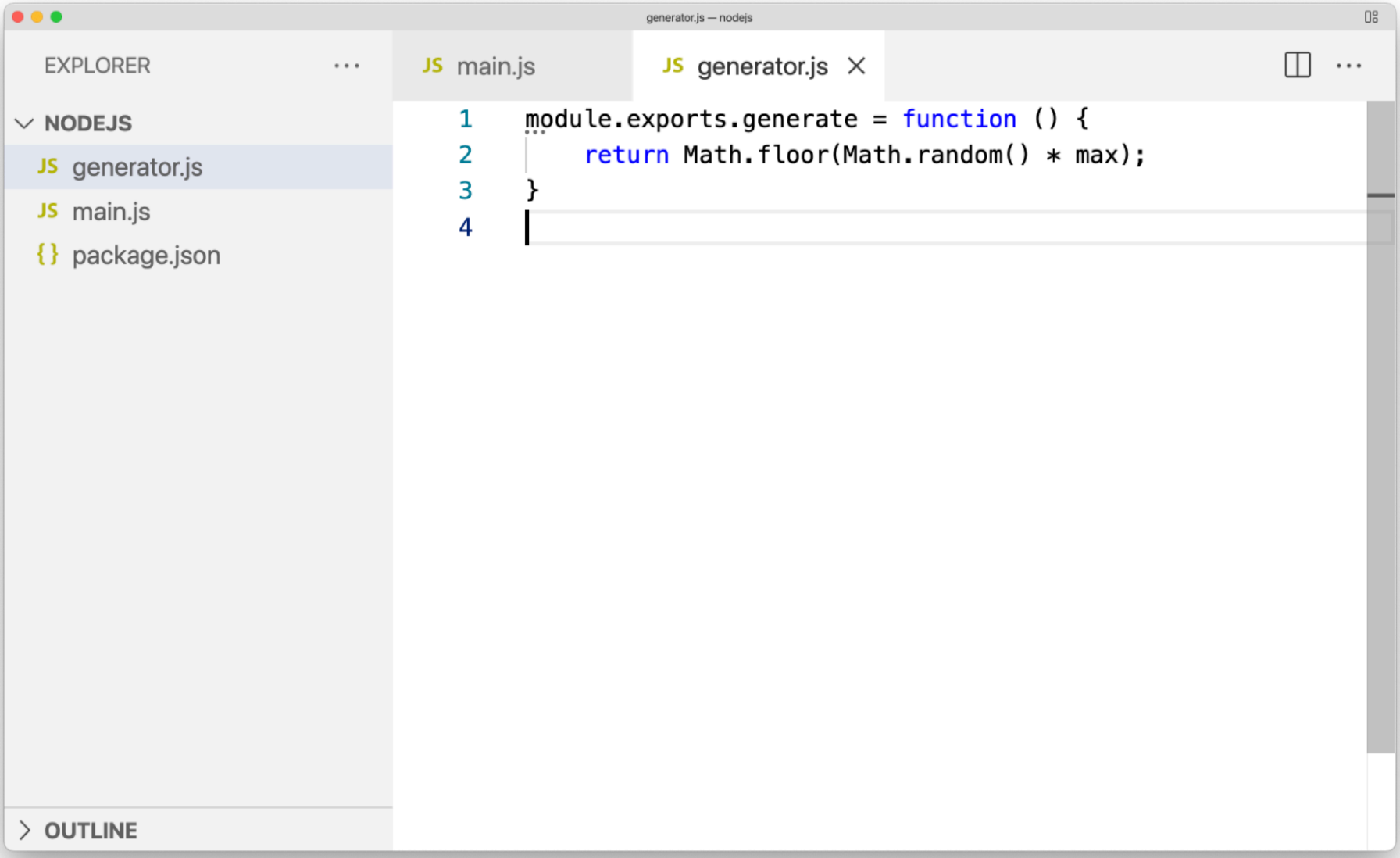
Onde estão as operações require e os objetos module e exports?

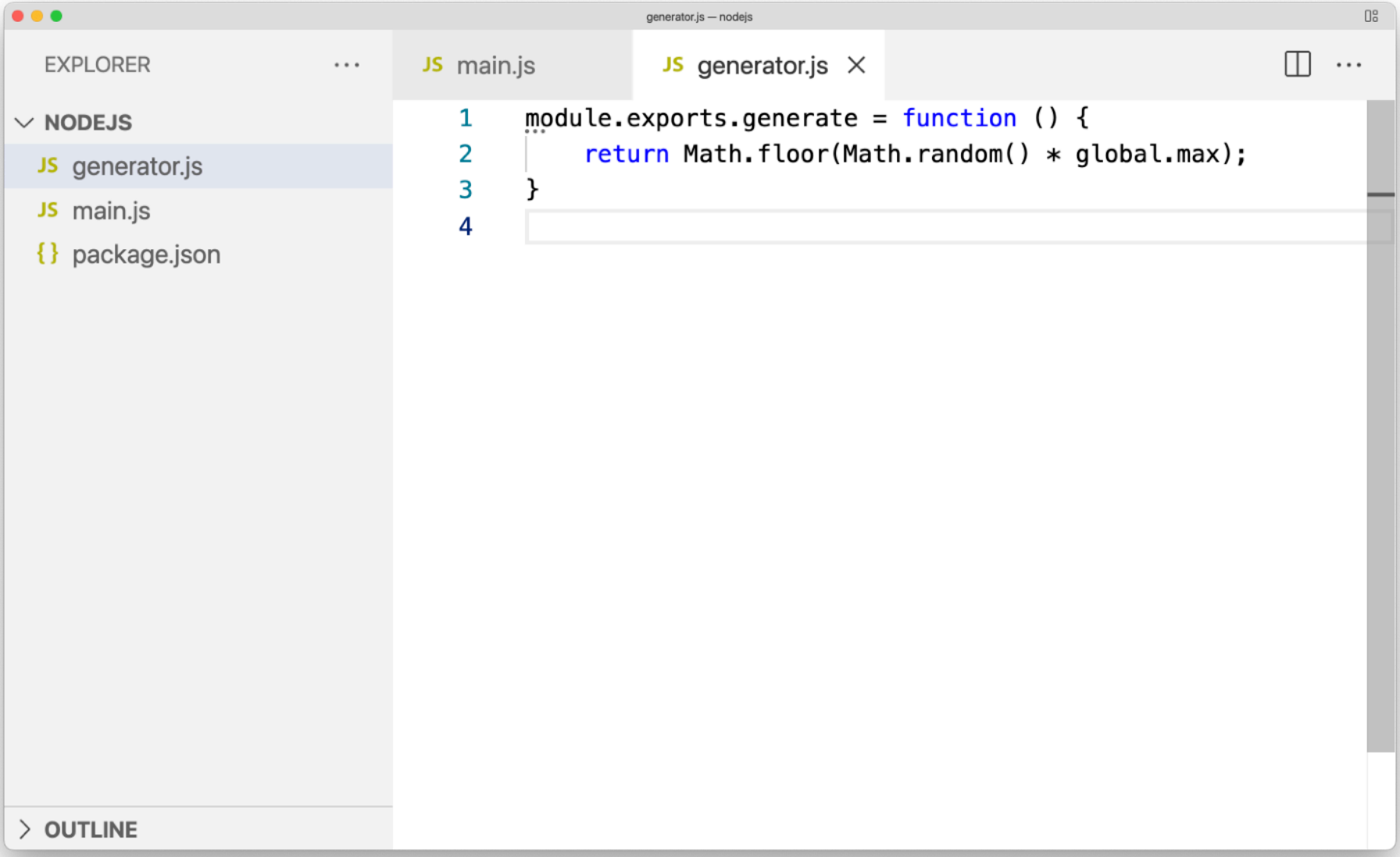


https://github.com/nodejs/node-v0.x-archive/blob/master/src/node.js

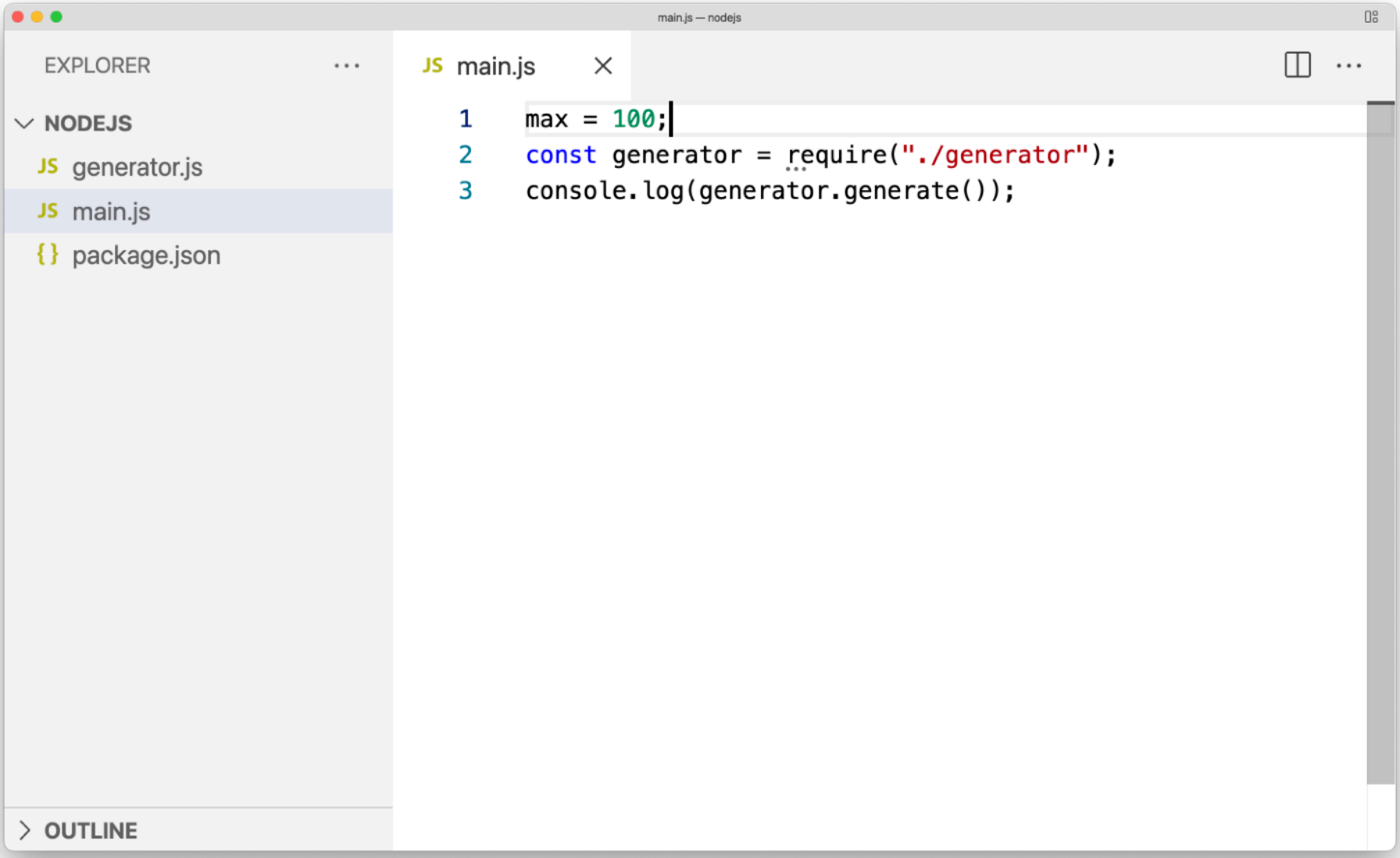
```
794  NativeModule.wrap = function(script) {
795    return NativeModule.wrapper[0] + script + NativeModule.wrapper[1];
796  };
797
798  NativeModule.wrapper = [
799    '(function (exports, require, module, __filename, __dirname) { ',
800    '\n});'
801  ];
802
803  NativeModule.prototype.compile = function() {
804    var source = NativeModule.getSource(this.id);
805    source = NativeModule.wrap(source);
806
807    var fn = runInThisContext(source, { filename: this.filename });
808    fn(this.exports, NativeModule.require, this, this.filename);
809
810    this.loaded = true;
811  };
812
```


main.js — nodejs
EXPLORER
NODEJS
generator.js
main.js
package.json
OUTLINE
main.js
generator.js
1 global.max = 100;
2 const generator = require("./generator");
3 console.log(generator.generate());
PROBLEMS
OUTPUT
TERMINAL
zsh
rodrigobranas nodejs $ node main.js
95
rodrigobranas nodejs $

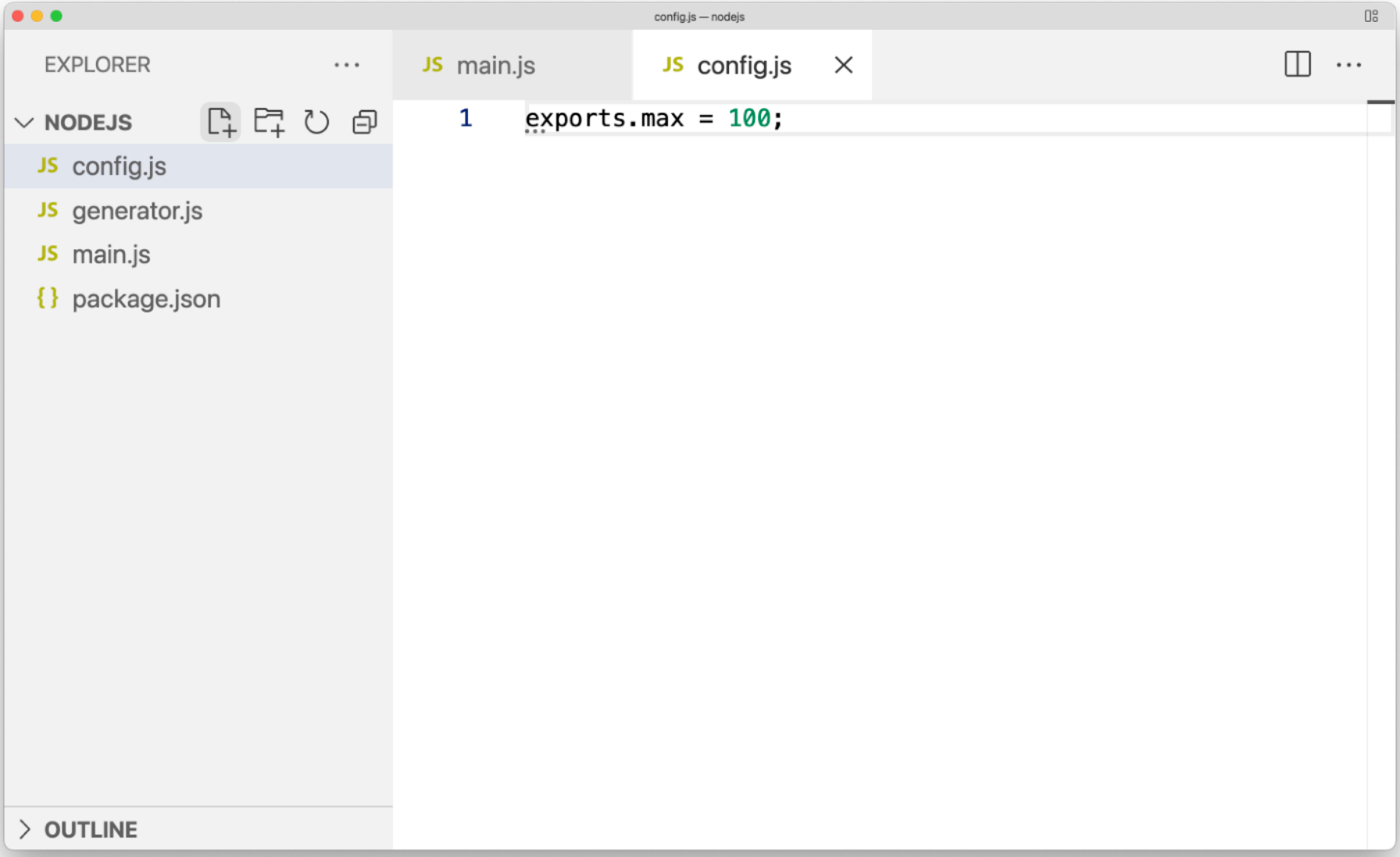

generator.js — nodejs
EXPLORER
NODEJS
generator.js
main.js
package.json
OUTLINE
main.js
generator.js
1 module.exports.generate = function () {
2 return Math.floor(Math.random() * max);
3 }
4

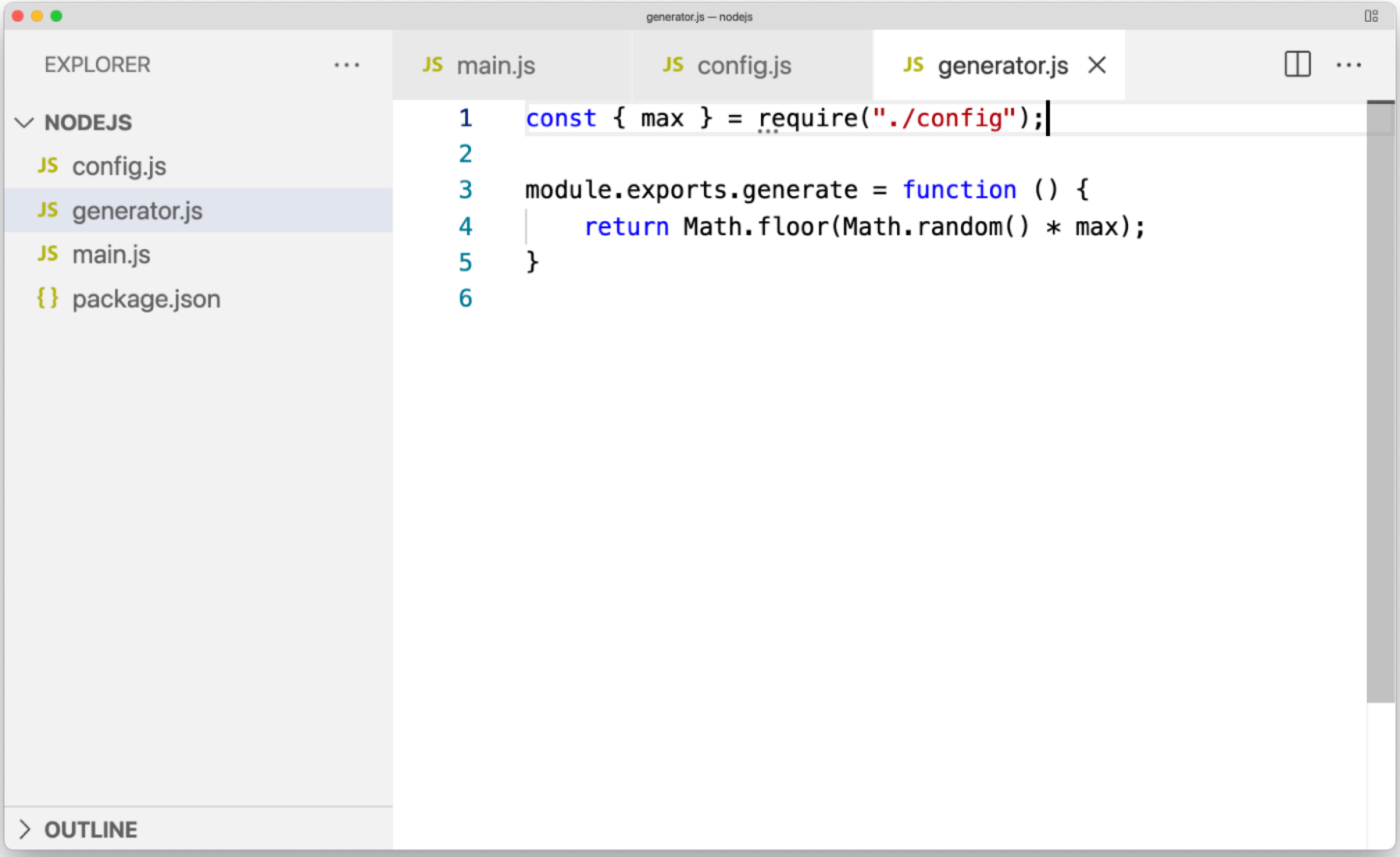

generator.js — nodejs
EXPLORER
NODEJS
generator.js
main.js
package.json
OUTLINE
main.js
generator.js
1 module.exports.generate = function () {
2 return Math.floor(Math.random() * global.max);
3 }
4

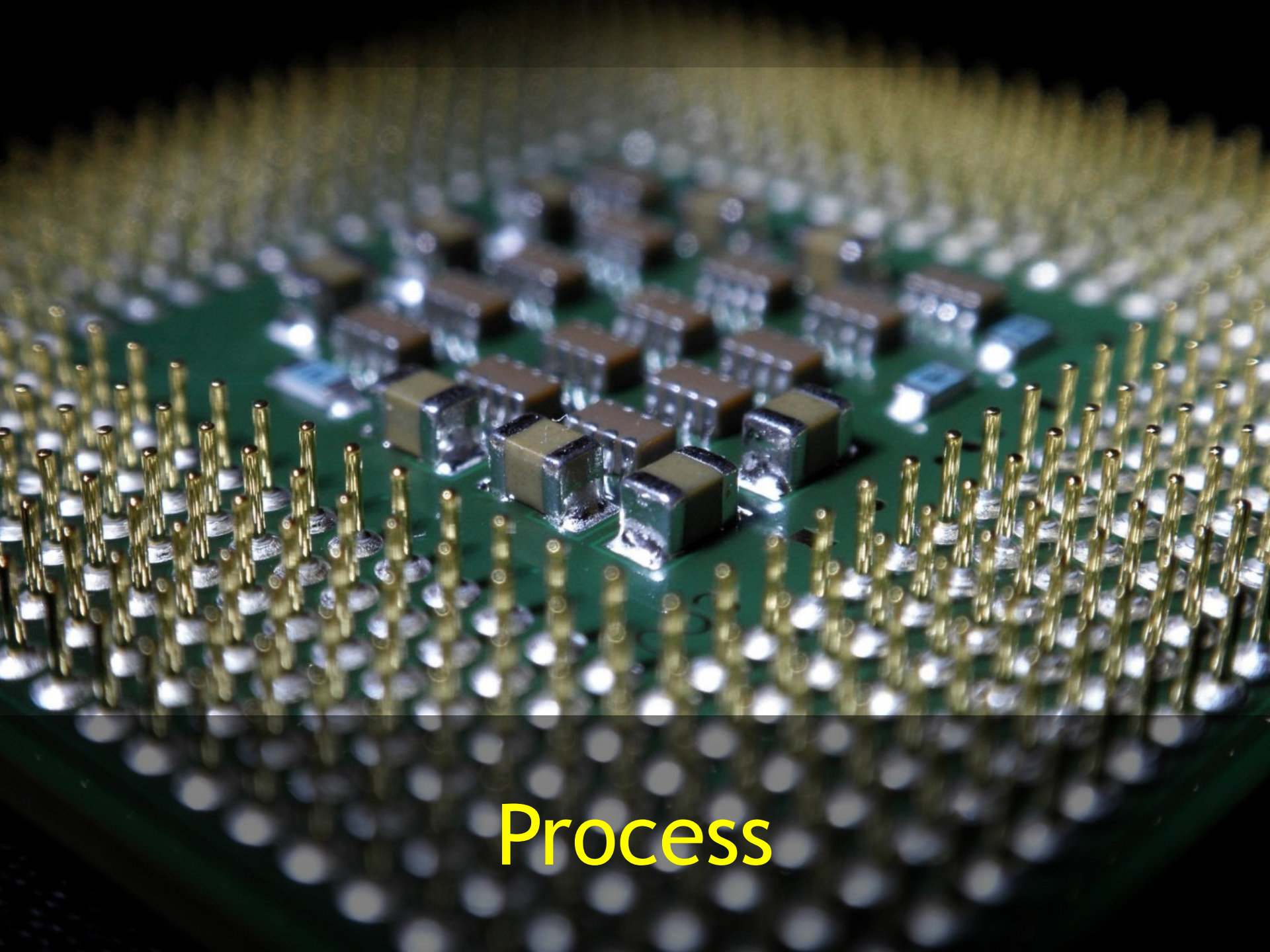
É possível criar uma variável global sem utilizar const, let e var

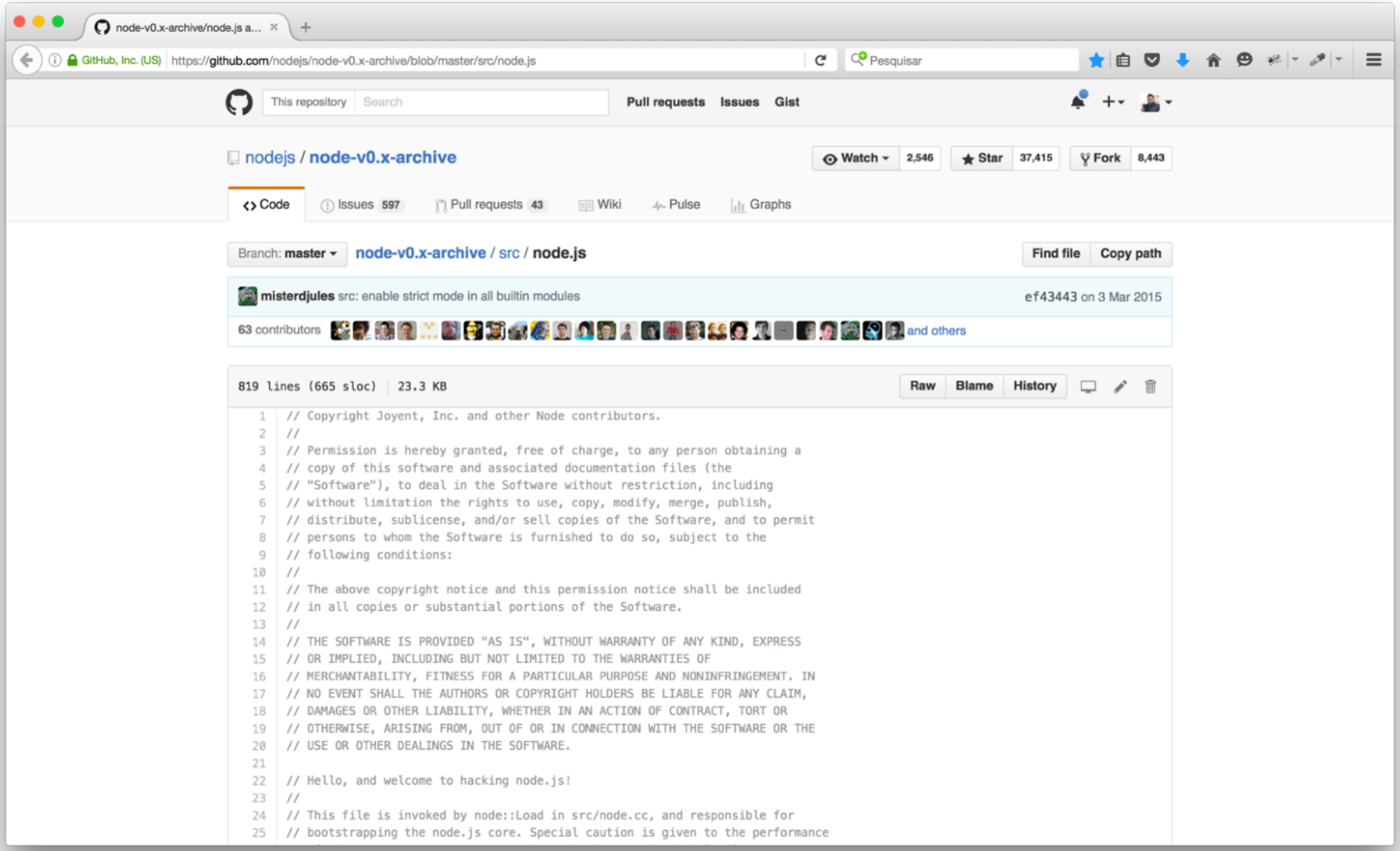

main.js — nodejs
EXPLORER
NODEJS
generator.js
main.js
package.json
OUTLINE
main.js
1 max = 100;
2 const generator = require("./generator");
3 console.log(generator.generate());


Como fazer para criar variáveis
globais sem utilizar o escopo global?


config.js — nodejs
EXPLORER
NODEJS
config.js
generator.js
main.js
package.json
OUTLINE
main.js
config.js
1
exports.max = 100;

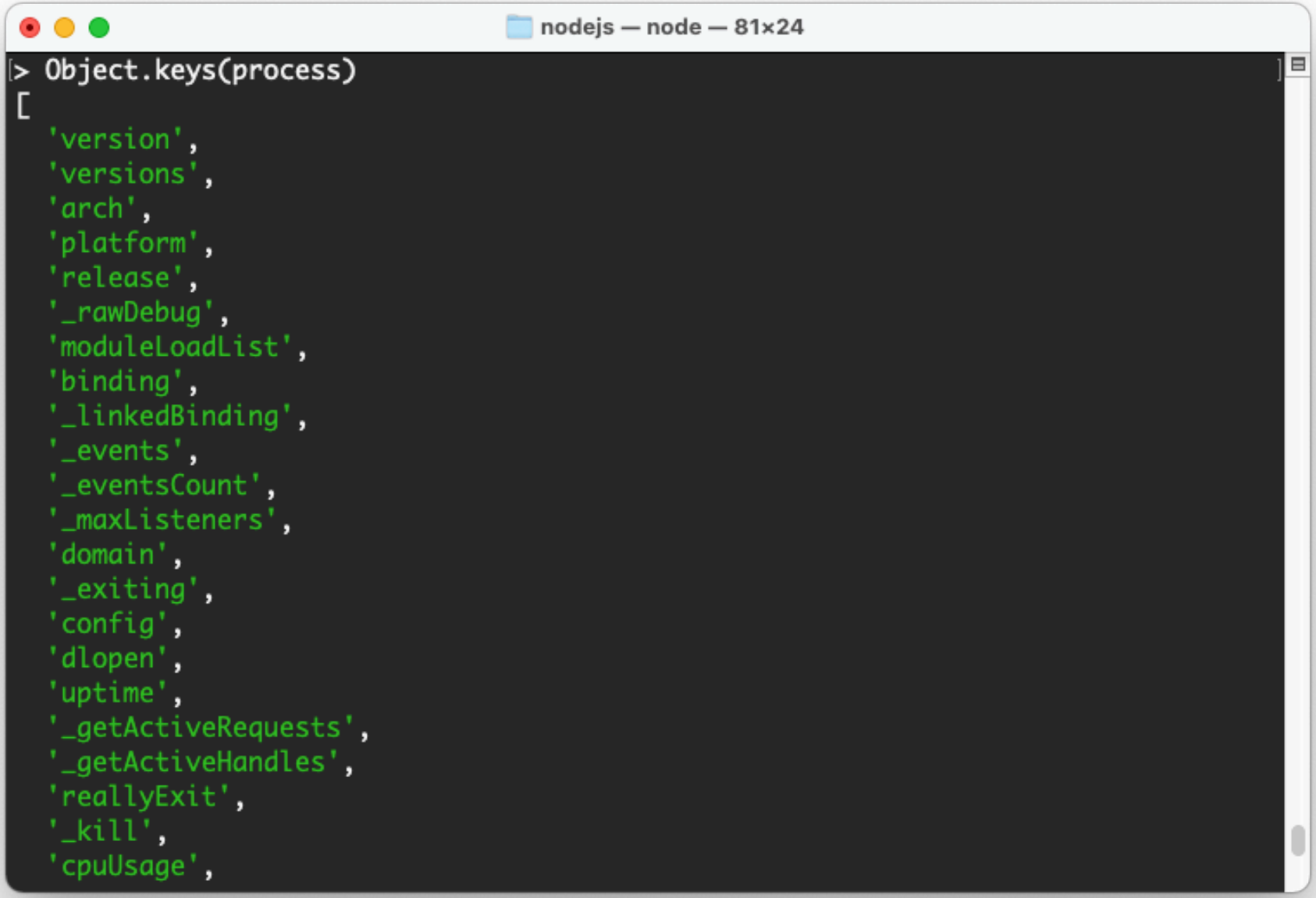

generator.js — nodejs
EXPLORER
NODEJS
config.js
generator.js
main.js
package.json
OUTLINE
main.js
config.js
generator.js
const { max } = require("./config");
module.exports.generate = function () {
return Math.floor(Math.random() * max);
}

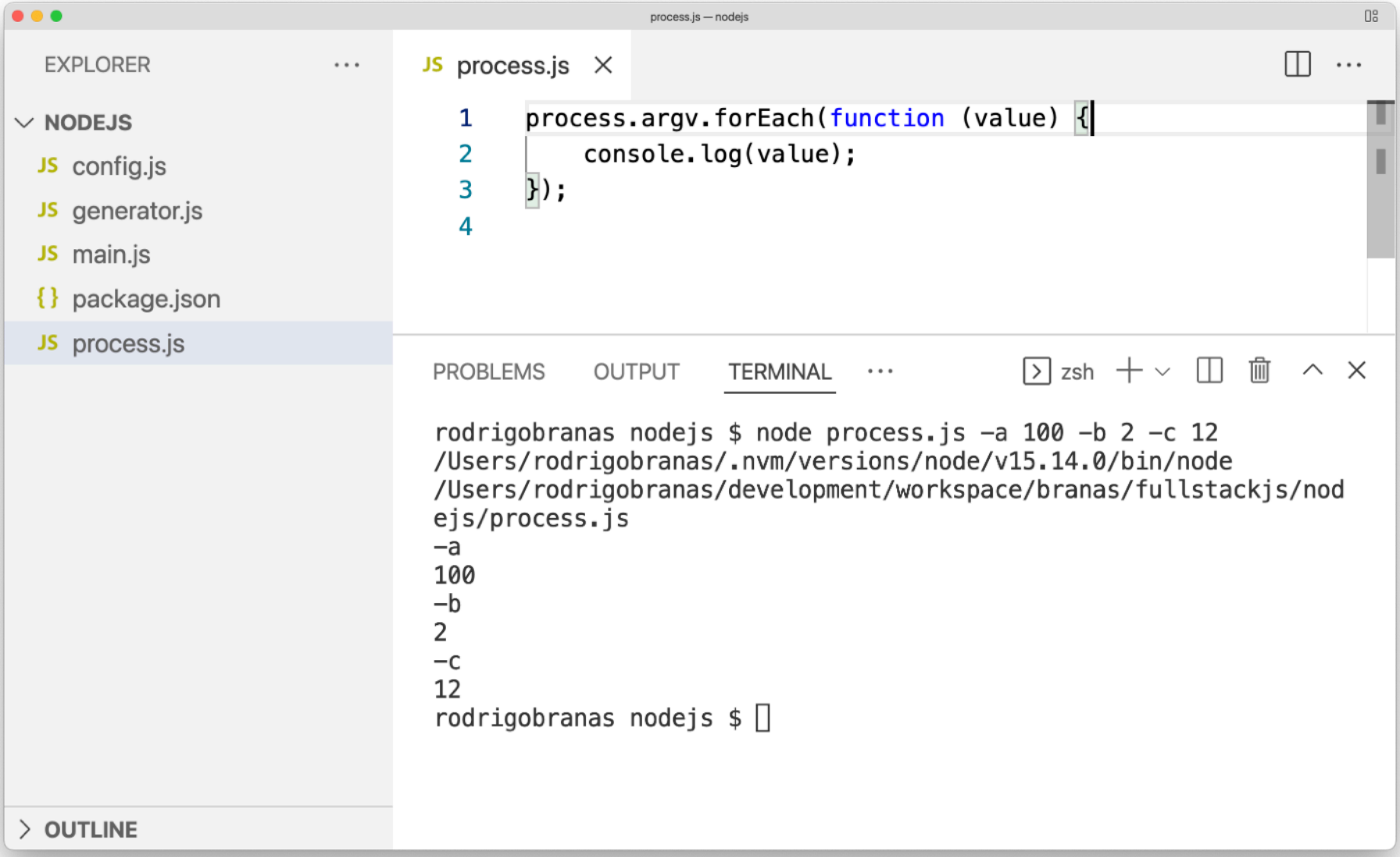

Process



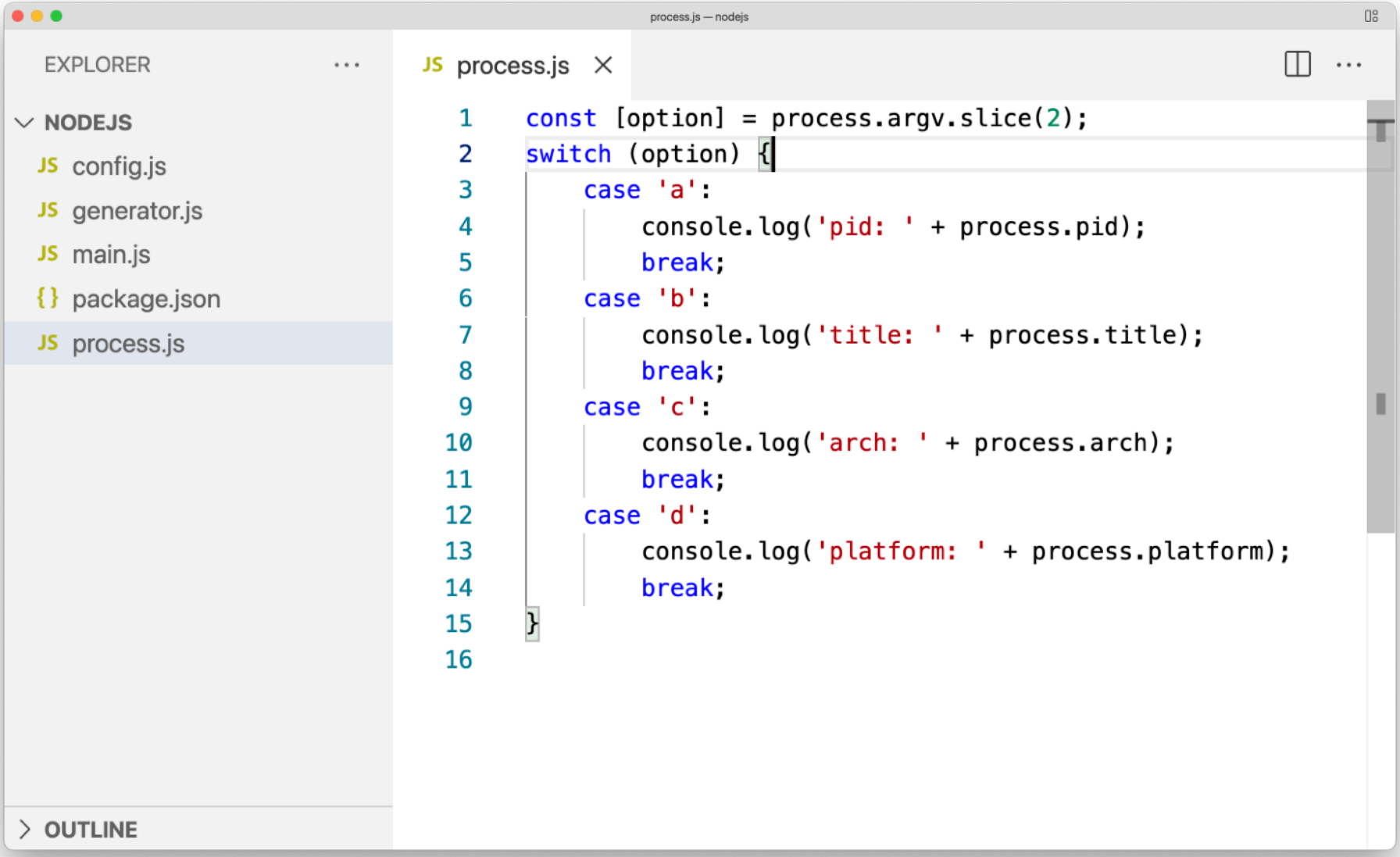
https://github.com/nodejs/node-v0.x-archive/blob/master/src/node.js

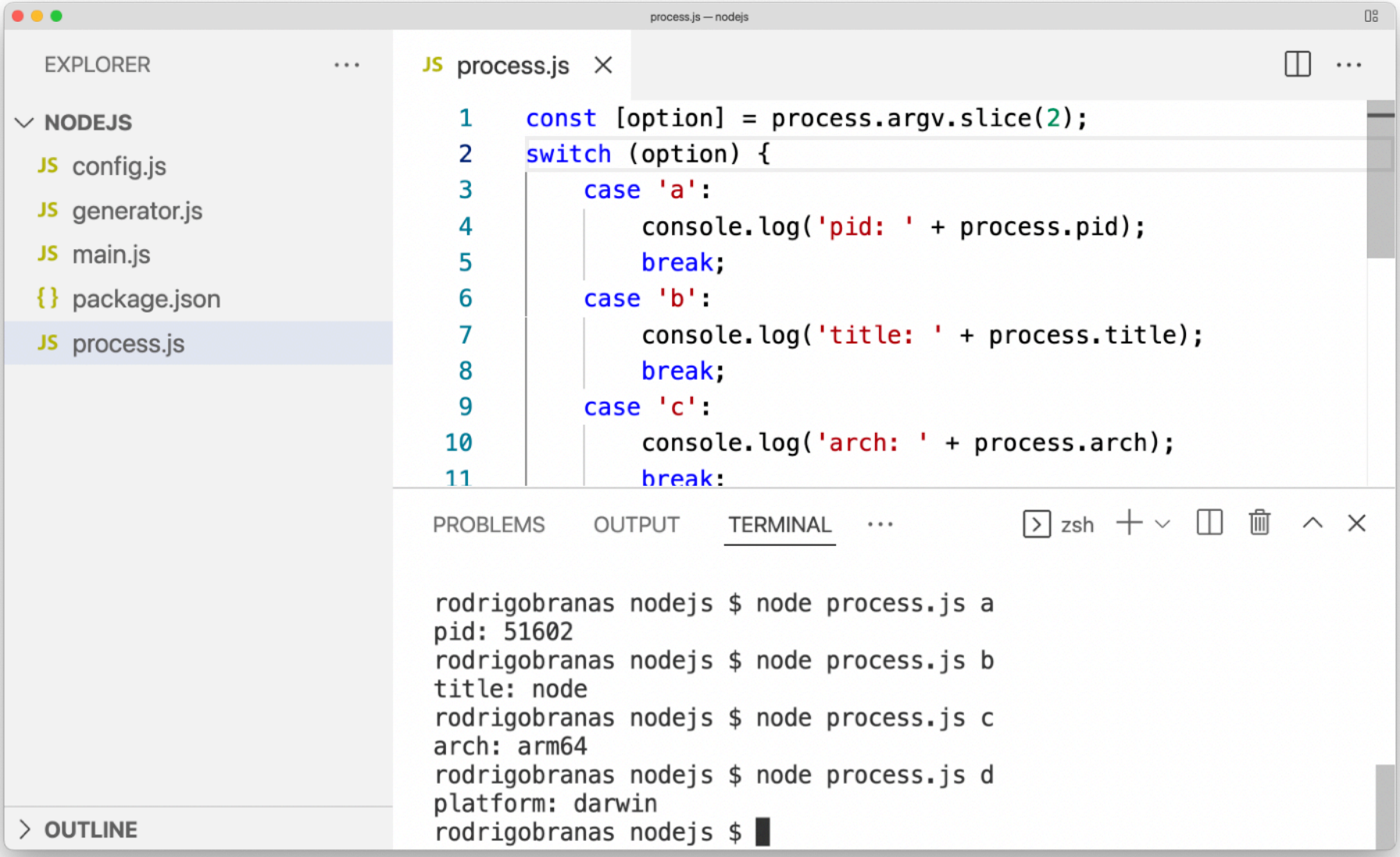
```
172
173   startup.globalVariables = function() {
174     global.process = process;
175     global.global = global;
176     global.GLOBAL = global;
177     global.root = global;
178     global.Buffer = NativeModule.require('buffer').Buffer;
179     process.domain = null;
180     process._exiting = false;
181   };
182
```


nodejs — node — 81×24
> Object.keys(process)
[
'version',
'versions',
'arch',
'platform',
'release',
'_rawDebug',
'moduleLoadList',
'binding',
'_linkedBinding',
'_events',
'_eventsCount',
'_maxListeners',
'domain',
'_exiting',
'config',
'dlopen',
'uptime',
'_getActiveRequests',
'_getActiveHandles',
'reallyExit',
'_kill',
'cpuUsage',


process.js — nodejs
EXPLORER
NODEJS
config.js
generator.js
main.js
package.json
process.js
OUTLINE
process.js
1 process.argv.forEach(function (value) {
2 console.log(value);
3 });
4
PROBLEMS
OUTPUT
TERMINAL
zsh
rodrigobranas nodejs $ node process.js -a 100 -b 2 -c 12
/Users/rodrigobranas/.nvm/versions/node/v15.14.0/bin/node
/Users/rodrigobranas/development/workspace/branas/fullstackjs/nodejs/process.js
-a
100
-b
2
-c
12
rodrigobranas nodejs $

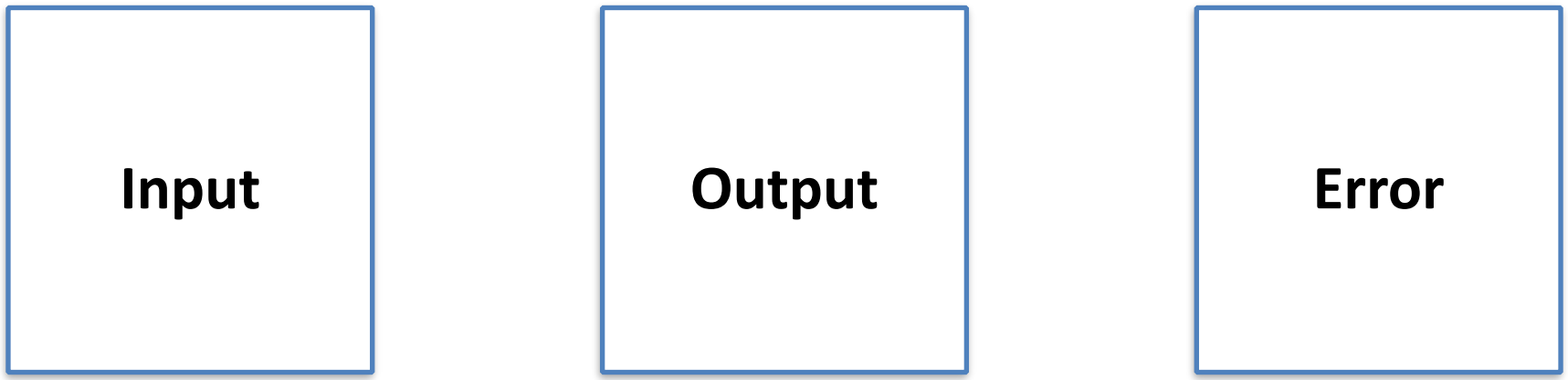
Exibindo informações do processo

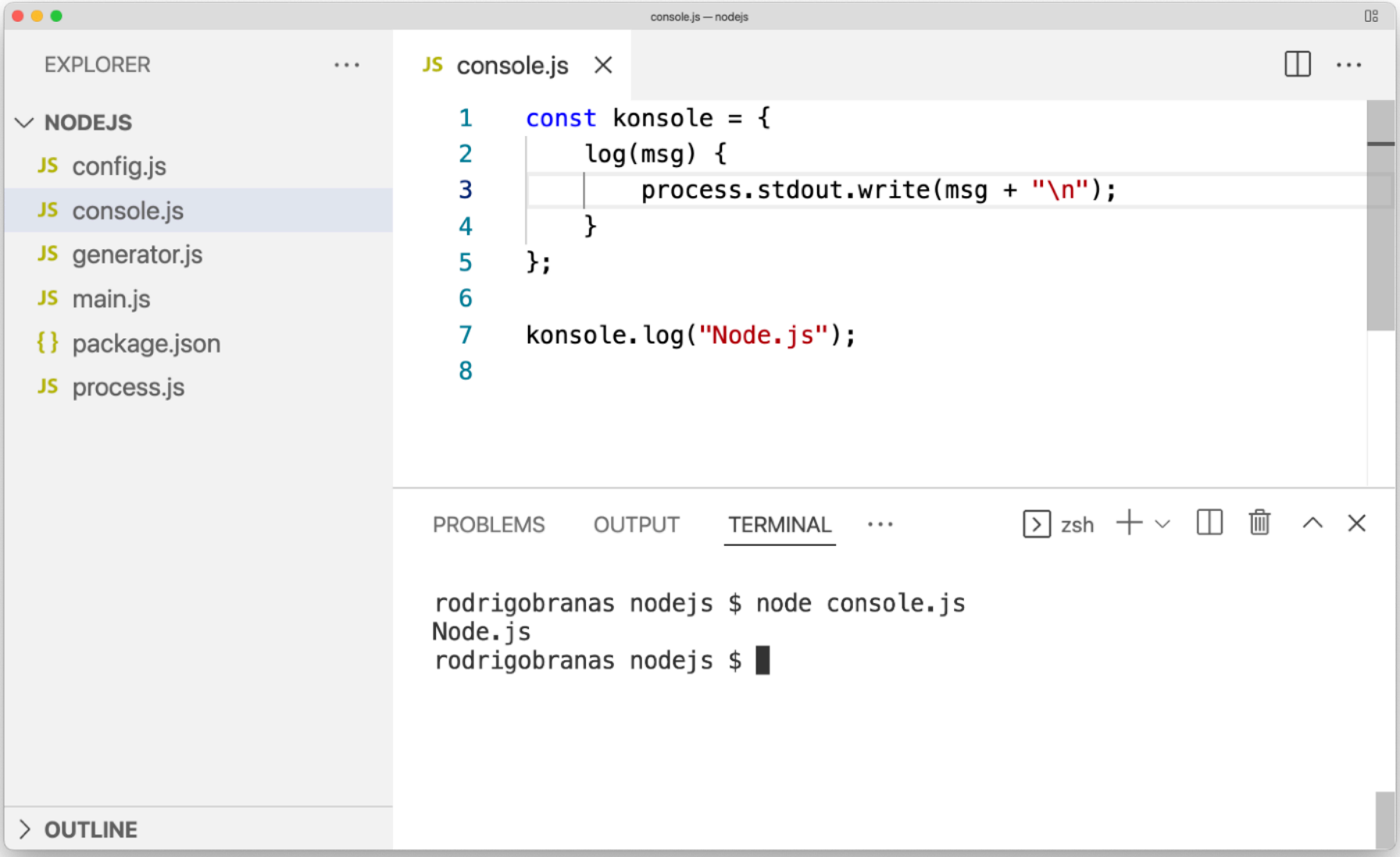

process.js — nodejs
EXPLORER
NODEJS
config.js
generator.js
main.js
package.json
process.js
OUTLINE
process.js
const [option] = process.argv.slice(2);
switch (option) {
case 'a':
console.log('pid: ' + process.pid);
break;
case 'b':
console.log('title: ' + process.title);
break;
case 'c':
console.log('arch: ' + process.arch);
break;
case 'd':
console.log('platform: ' + process.platform);
break;
}

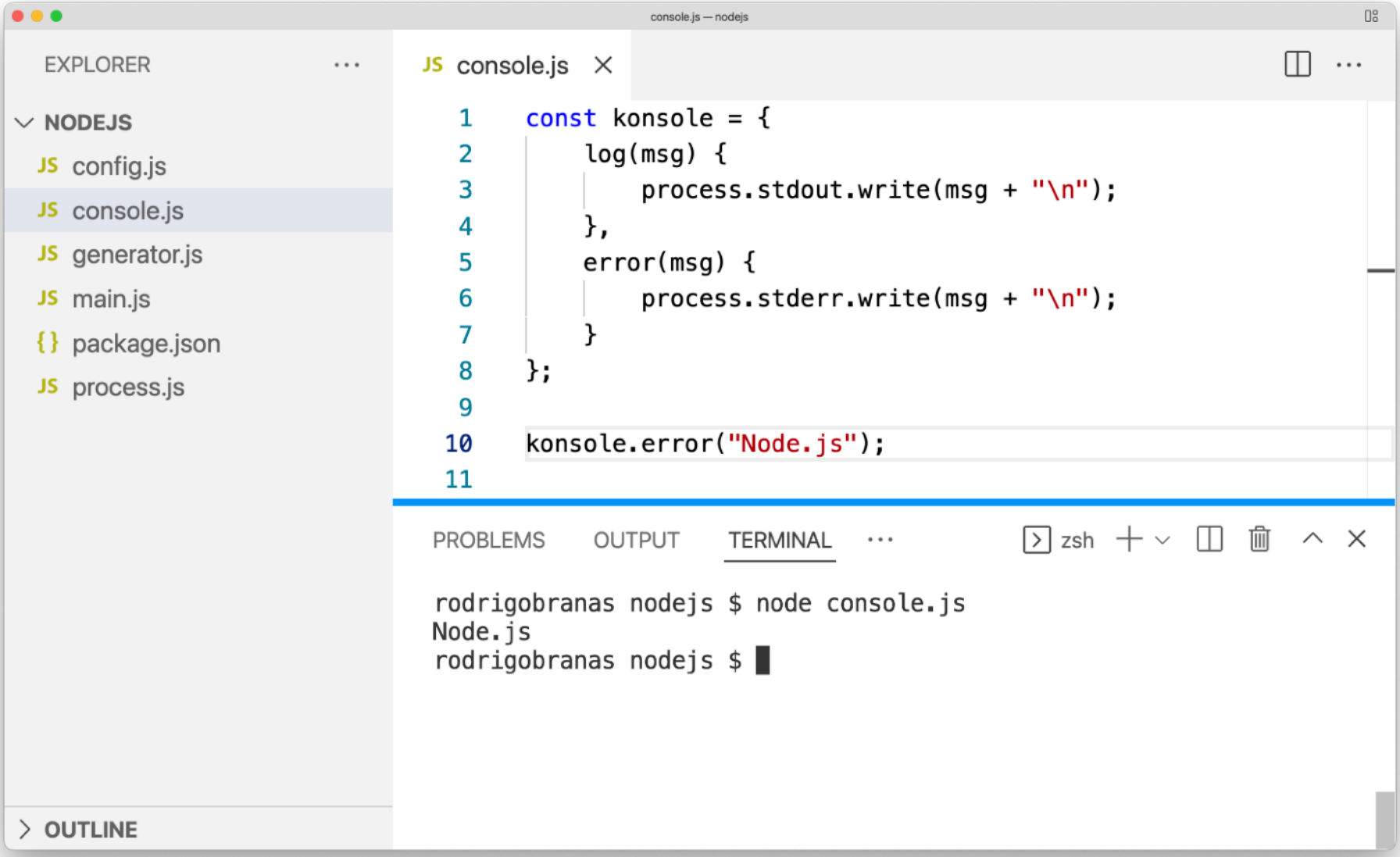

process.js — nodejs
EXPLORER
NODEJS
config.js
generator.js
main.js
package.json
process.js
OUTLINE
process.js
1 const [option] = process.argv.slice(2);
2 switch (option) {
3 case 'a':
4 console.log('pid: ' + process.pid);
5 break;
6 case 'b':
7 console.log('title: ' + process.title);
8 break;
9 case 'c':
10 console.log('arch: ' + process.arch);
11 break;
PROBLEMS
OUTPUT
TERMINAL
zsh
rodrigobranas nodejs $ node process.js a
pid: 51602
rodrigobranas nodejs $ node process.js b
title: node
rodrigobranas nodejs $ node process.js c
arch: arm64
rodrigobranas nodejs $ node process.js d
platform: darwin
rodrigobranas nodejs $


Standard Streams

São canais de comunicação, utilizados para realizar operações de entrada e saída, entre o programa e o ambiente onde ele está sendo executado.

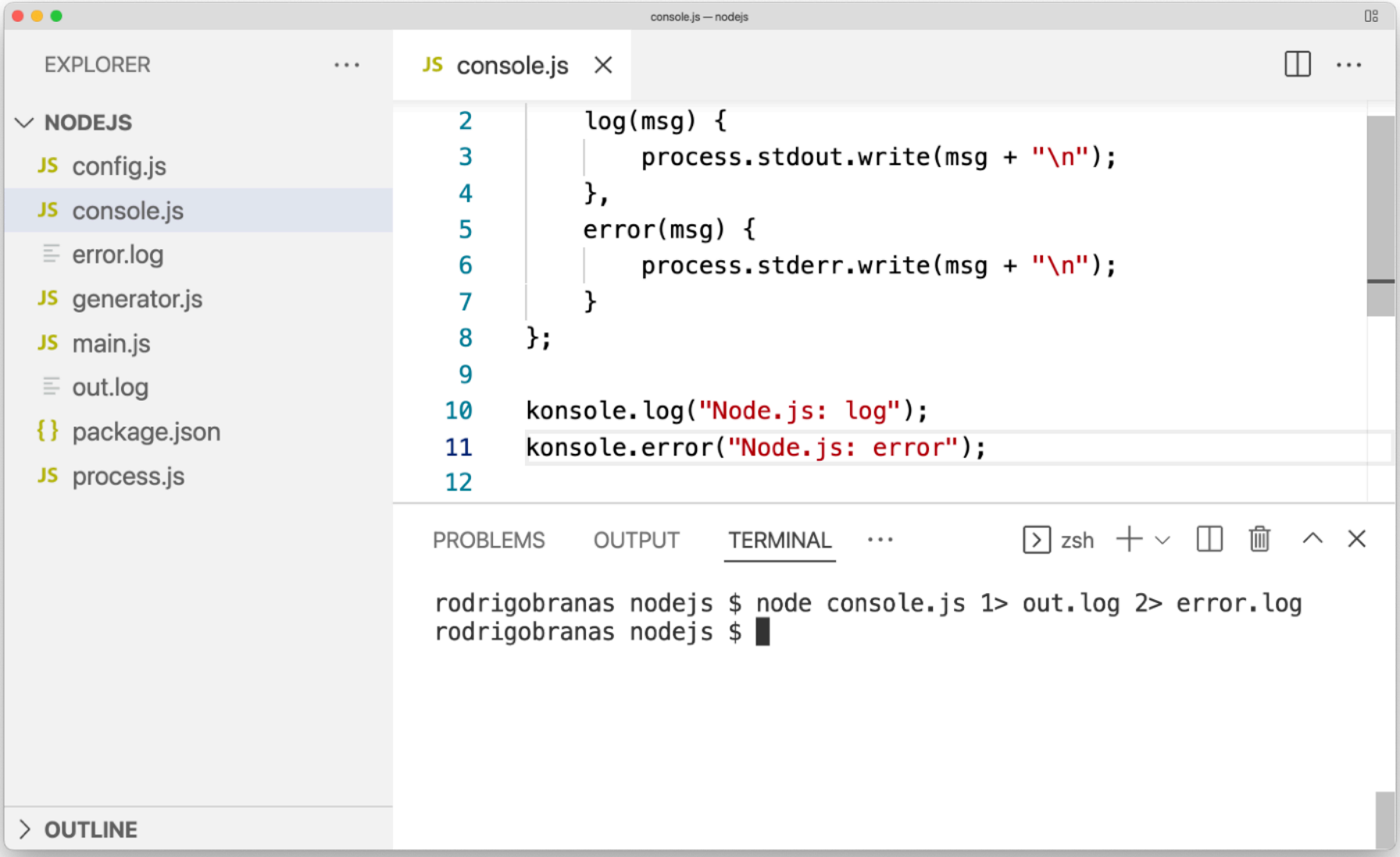

Input
Output
Error

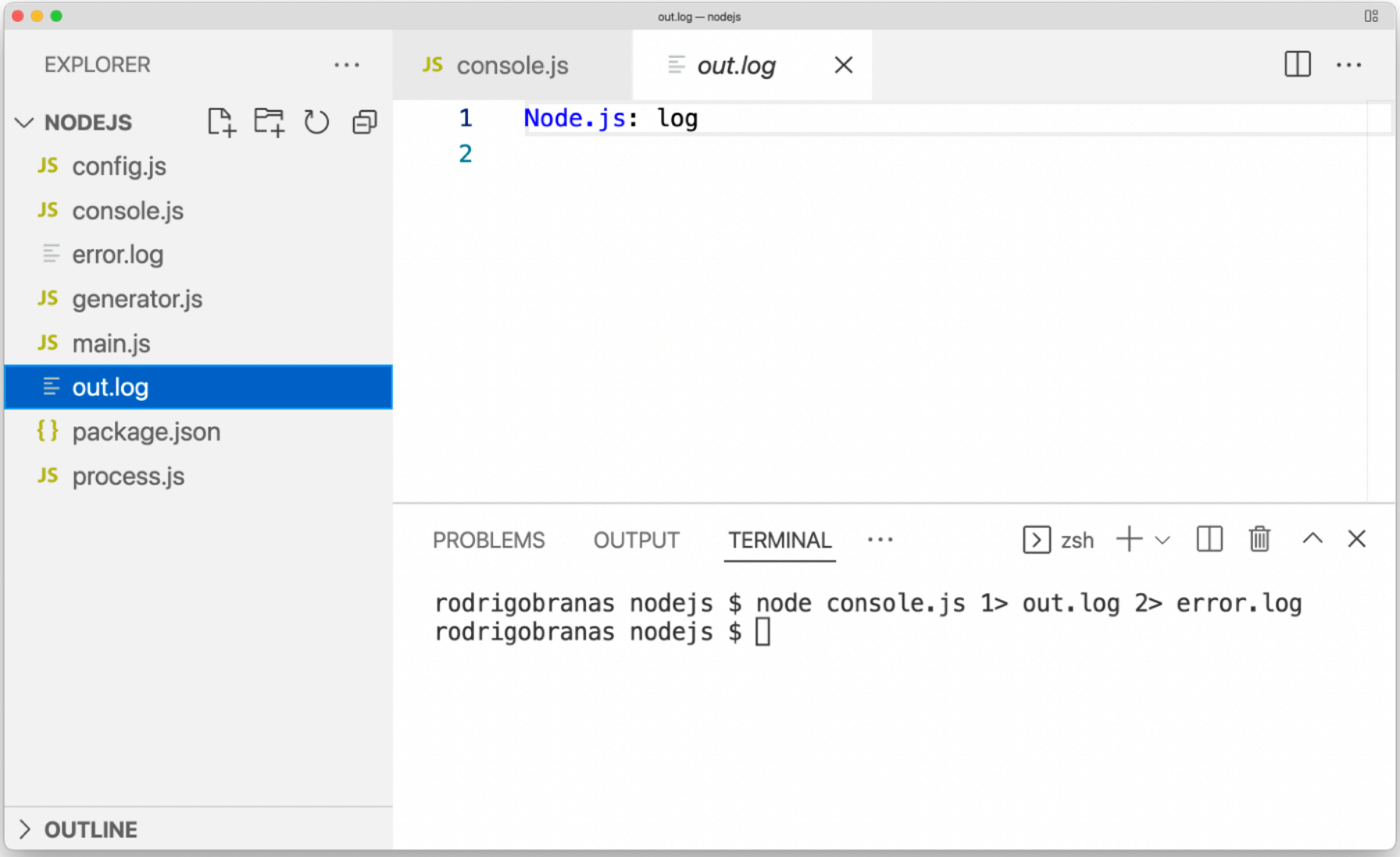

console.js — nodejs
EXPLORER
NODEJS
config.js
console.js
generator.js
main.js
package.json
process.js
OUTLINE
console.js
const konsole = {
log(msg) {
process.stdout.write(msg + "\n");
}
};
konsole.log("Node.js");
PROBLEMS
OUTPUT
TERMINAL
zsh
rodrigobranas nodejs $ node console.js
Node.js
rodrigobranas nodejs $

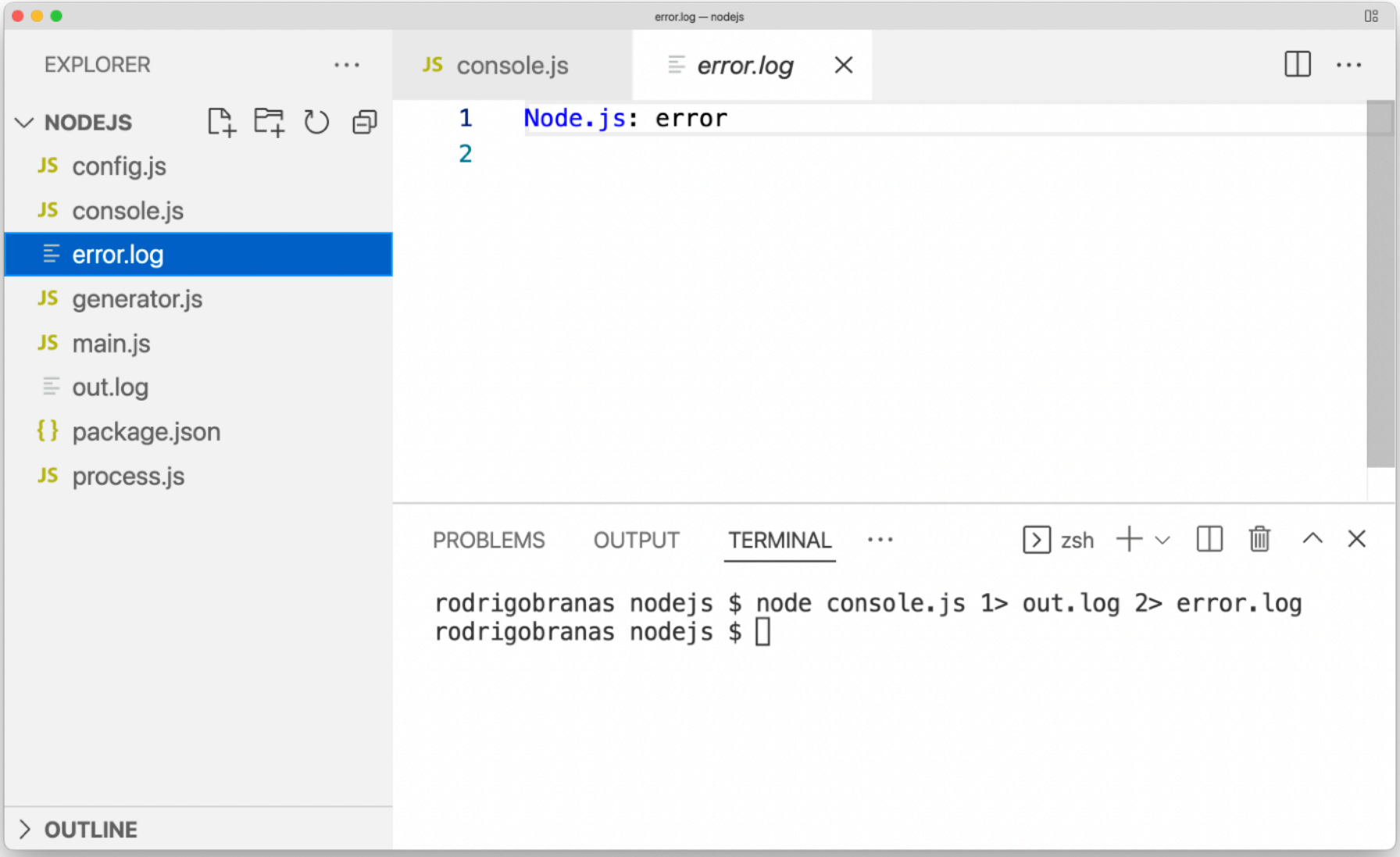

console.js — nodejs
EXPLORER
NODEJS
config.js
console.js
generator.js
main.js
package.json
process.js
OUTLINE
console.js
const konsole = {
log(msg) {
process.stdout.write(msg + "\n");
},
error(msg) {
process.stderr.write(msg + "\n");
}
};
konsole.error("Node.js");
PROBLEMS
OUTPUT
TERMINAL
zsh
rodrigobranas nodejs $ node console.js
Node.js
rodrigobranas nodejs $


Qual é a diferença entre output e error?

Eles podem ser **tratados** e **redirecionados** de formas diferentes

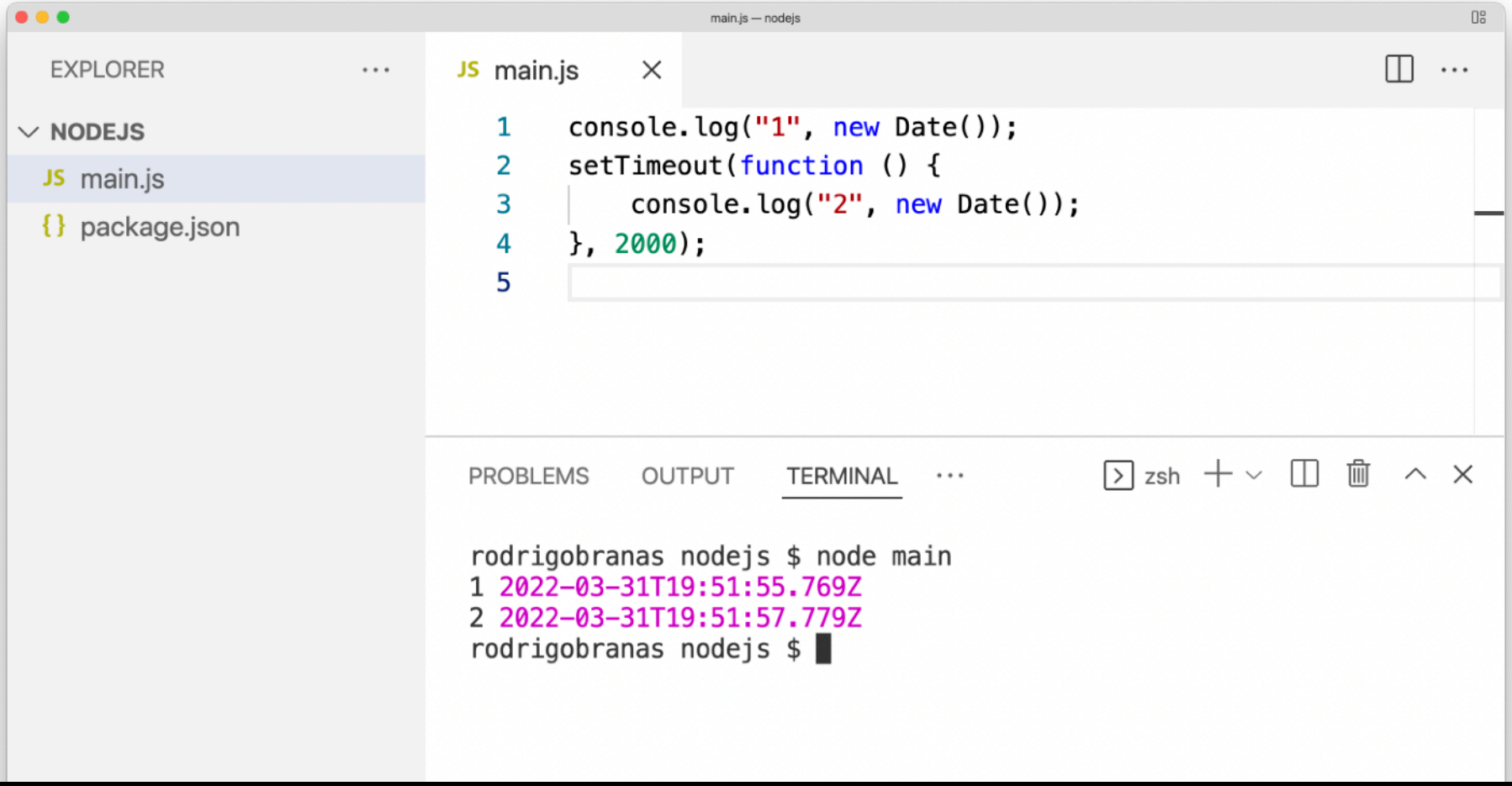

console.js — nodejs
EXPLORER
NODEJS
config.js
console.js
error.log
generator.js
main.js
out.log
package.json
process.js
OUTLINE
console.js
log(msg) {
process.stdout.write(msg + "\n");
},
error(msg) {
process.stderr.write(msg + "\n");
}
};
konsole.log("Node.js: log");
konsole.error("Node.js: error");
PROBLEMS
OUTPUT
TERMINAL
zsh
rodrigobranas nodejs $ node console.js 1> out.log 2> error.log
rodrigobranas nodejs $


out.log — nodejs
EXPLORER
NODEJS
config.js
console.js
error.log
generator.js
main.js
out.log
package.json
process.js
OUTLINE
console.js
out.log
1 Node.js: log
2
PROBLEMS
OUTPUT
TERMINAL
zsh
rodrigobranas nodejs $ node console.js 1> out.log 2> error.log
rodrigobranas nodejs $

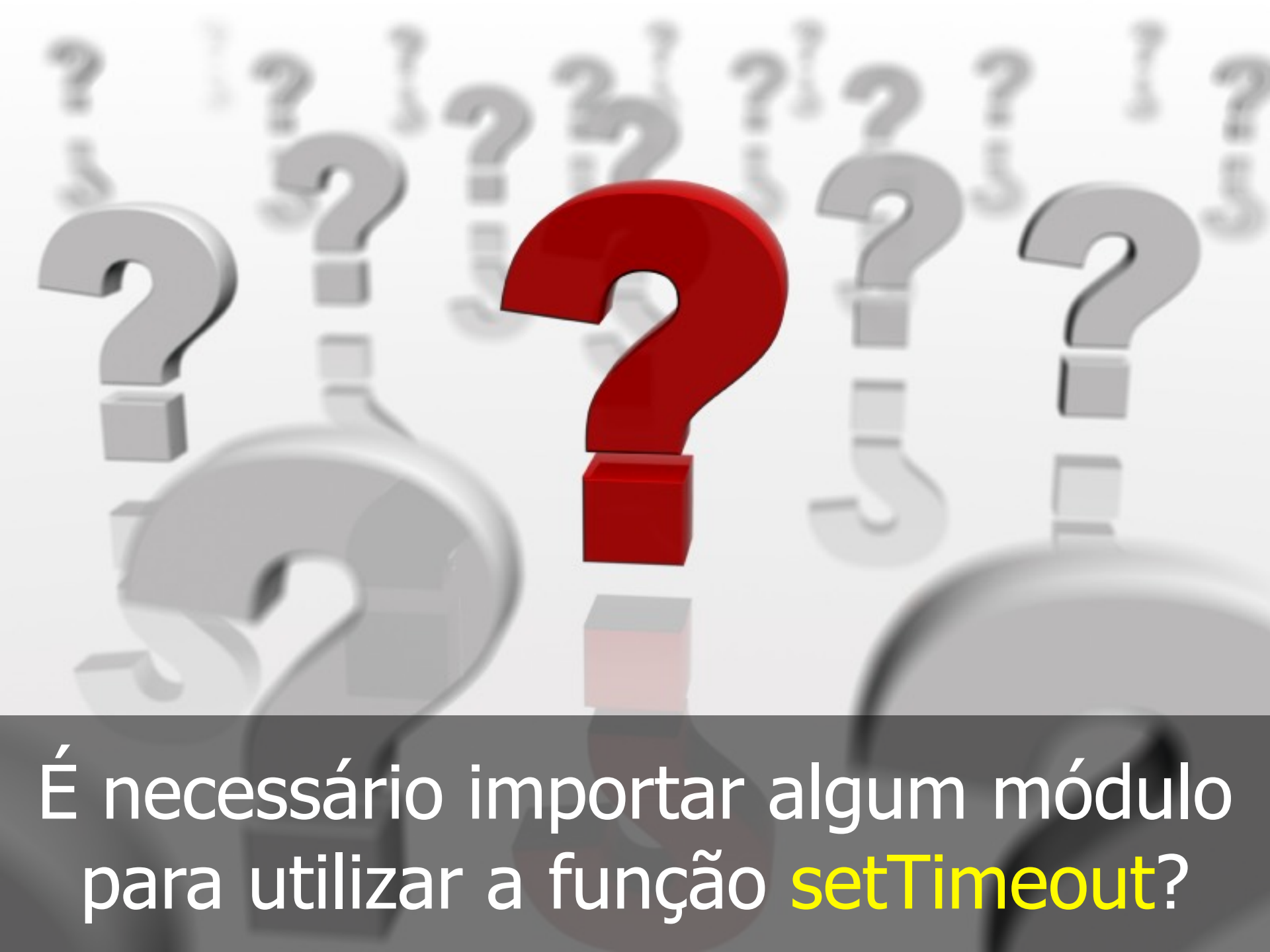

error.log — nodejs
EXPLORER
NODEJS
config.js
console.js
error.log
generator.js
main.js
out.log
package.json
process.js
OUTLINE
console.js
error.log
1 Node.js: error
2
PROBLEMS
OUTPUT
TERMINAL
zsh
rodrigobranas nodejs $ node console.js 1> out.log 2> error.log
rodrigobranas nodejs $

Timers

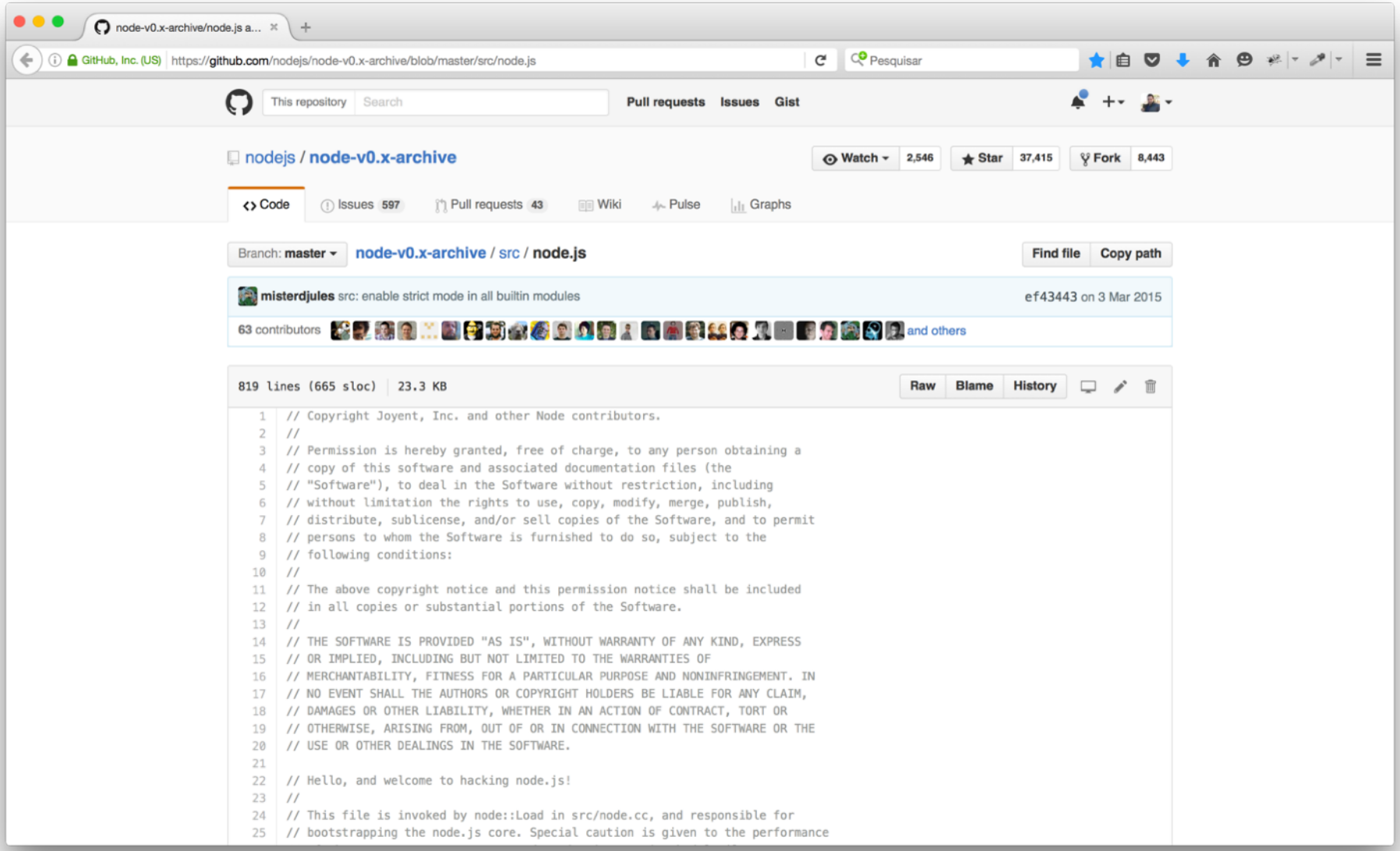
Por meio dos timers, é possível agendar a execução de funções



Criando um timer com setTimeout

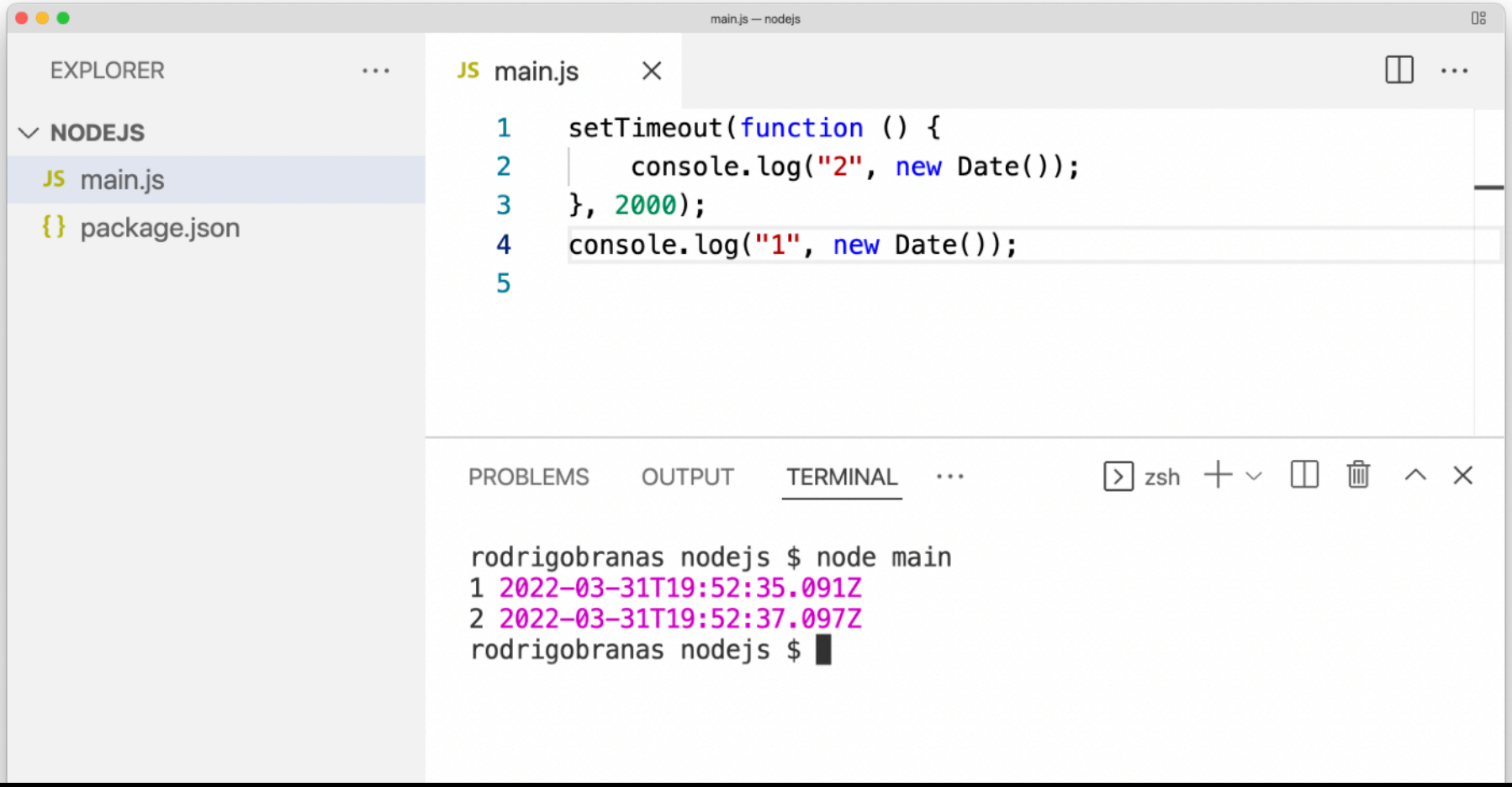


É necessário importar algum módulo para utilizar a função setTimeout?

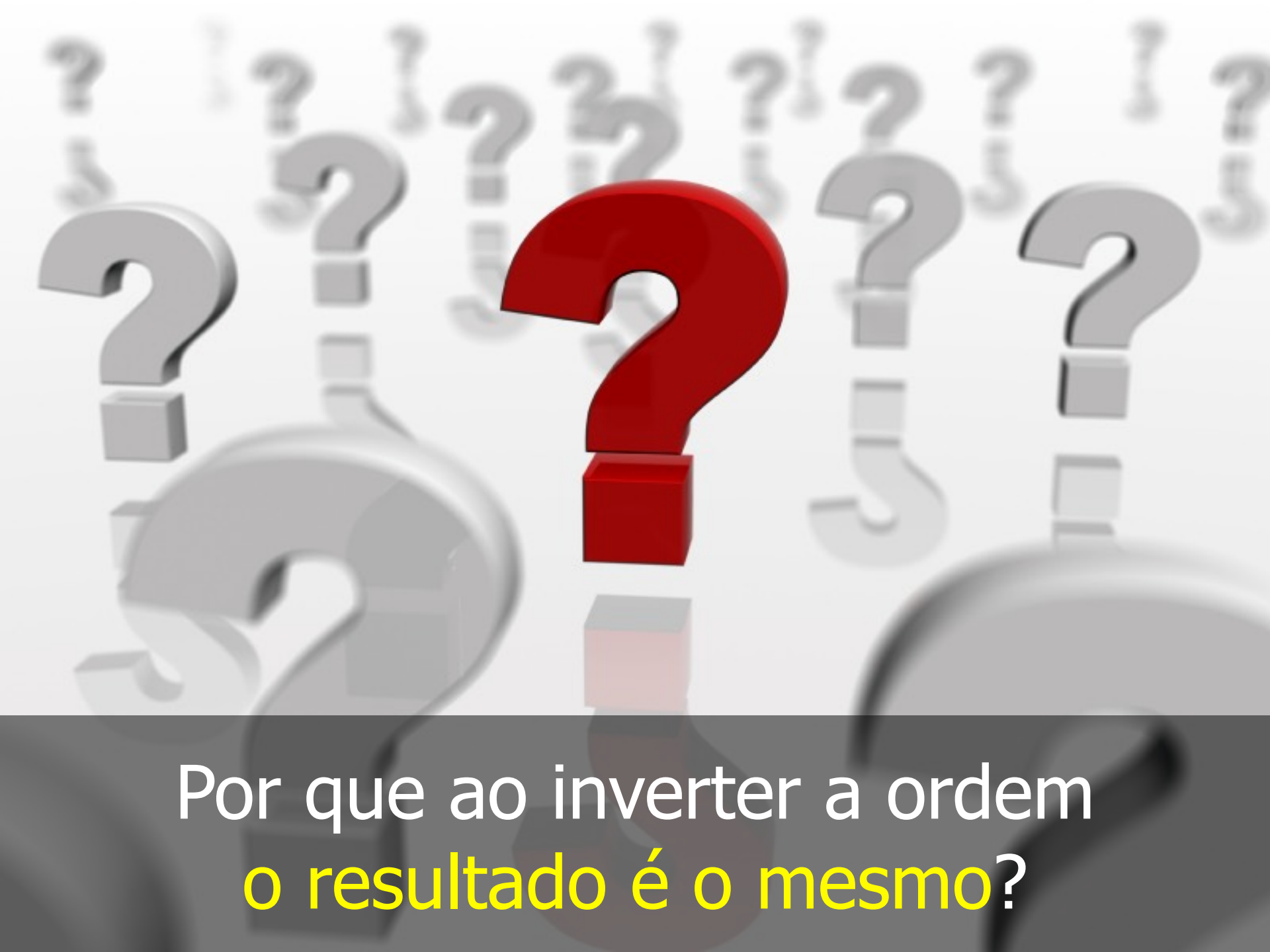


https://github.com/nodejs/node-v0.x-archive/blob/master/src/node.js

```
182
183   startup.globalTimeouts = function() {
184     global.setTimeout = function() {
185       var t = NativeModule.require('timers');
186       return t.setTimeout.apply(this, arguments);
187     };
188
189     global.setInterval = function() {
190       var t = NativeModule.require('timers');
191       return t.setInterval.apply(this, arguments);
192     };
193
194     global.clearTimeout = function() {
195       var t = NativeModule.require('timers');
196       return t.clearTimeout.apply(this, arguments);
197     };
198
199     global.clearInterval = function() {
200       var t = NativeModule.require('timers');
201       return t.clearInterval.apply(this, arguments);
202     };
203
204     global.setImmediate = function() {
205       var t = NativeModule.require('timers');
206       return t.setImmediate.apply(this, arguments);
207     };
208
209     global.clearImmediate = function() {
210       var t = NativeModule.require('timers');
211       return t.clearImmediate.apply(this, arguments);
212     };
213   };
```

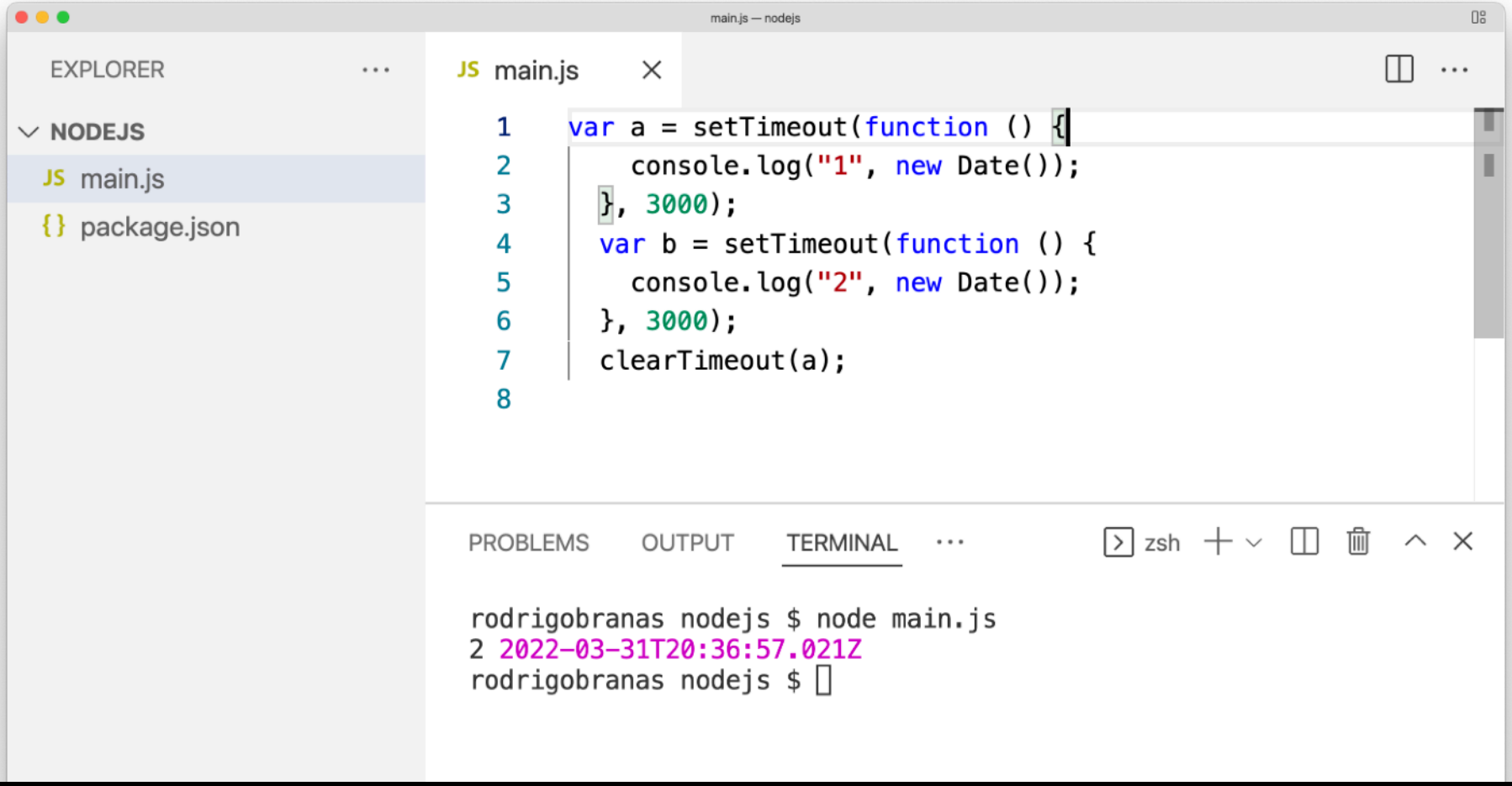


Invertendo a ordem de execução



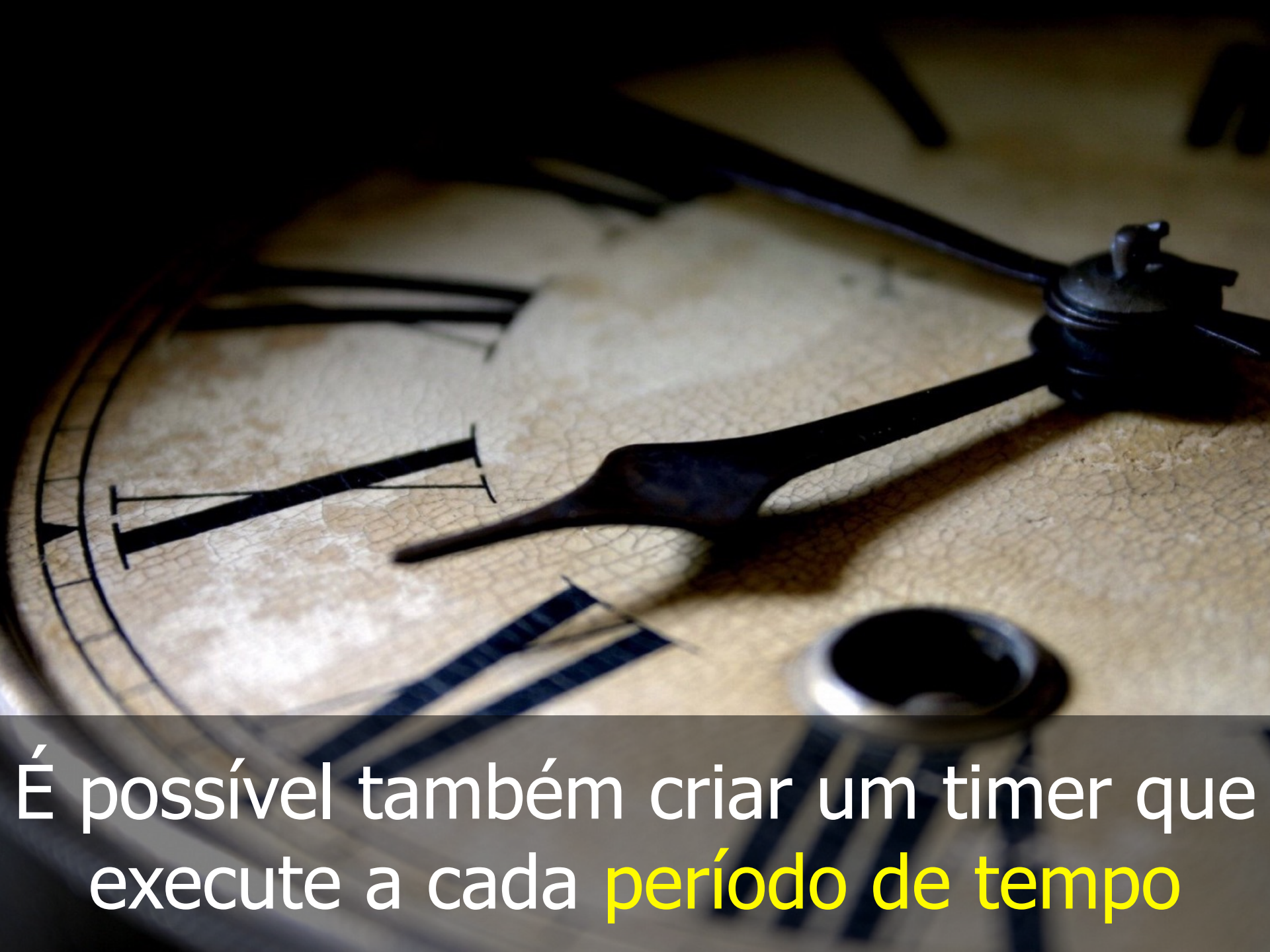
Por que ao inverter a ordem
o resultado é o mesmo?

Para evitar o bloqueio do Event Loop, que é executado em uma única thread, as funções invocadas pelos timers ficam aguardando na Event Queue

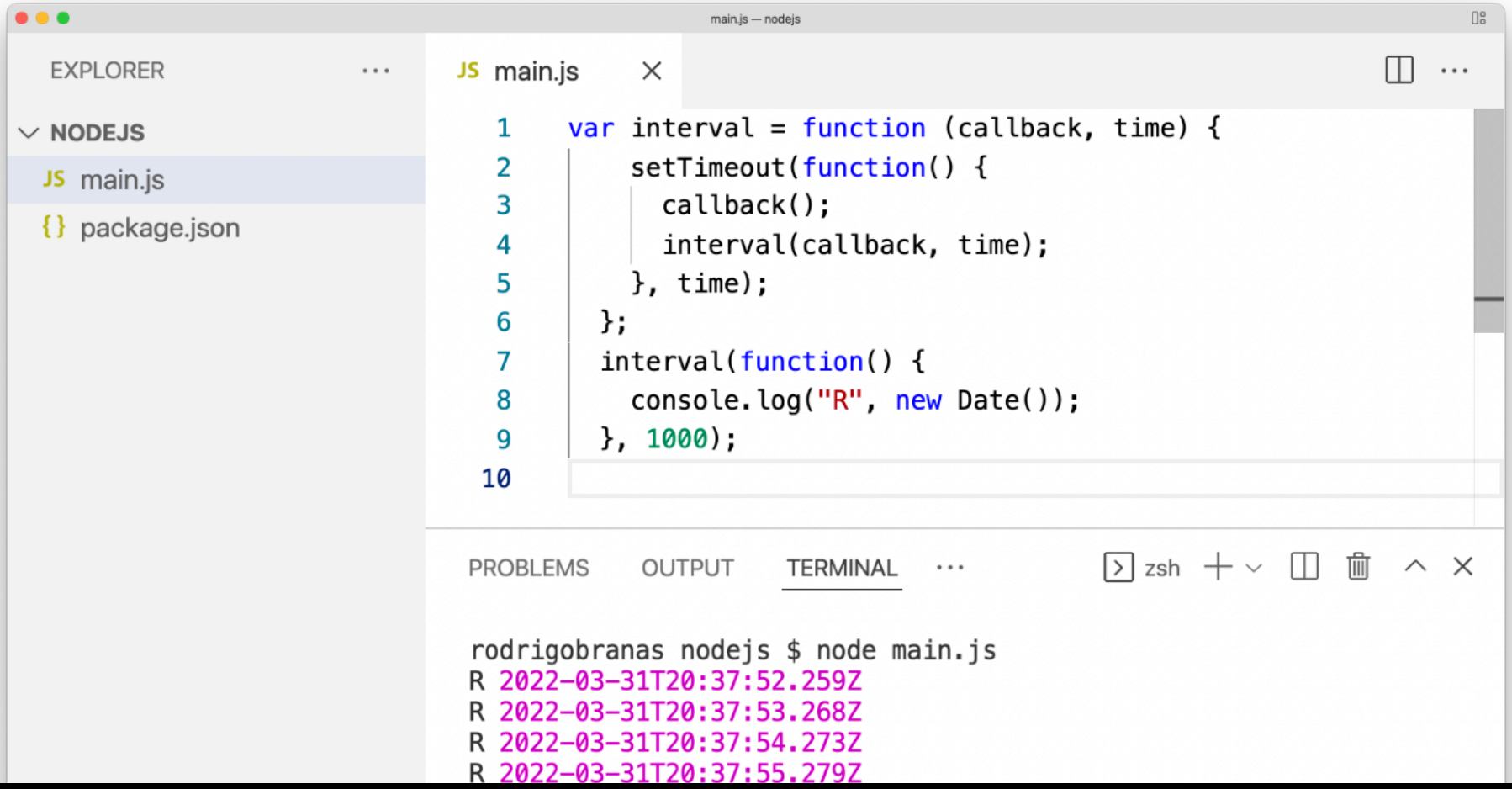


Cancelando um timer com clearTimeout

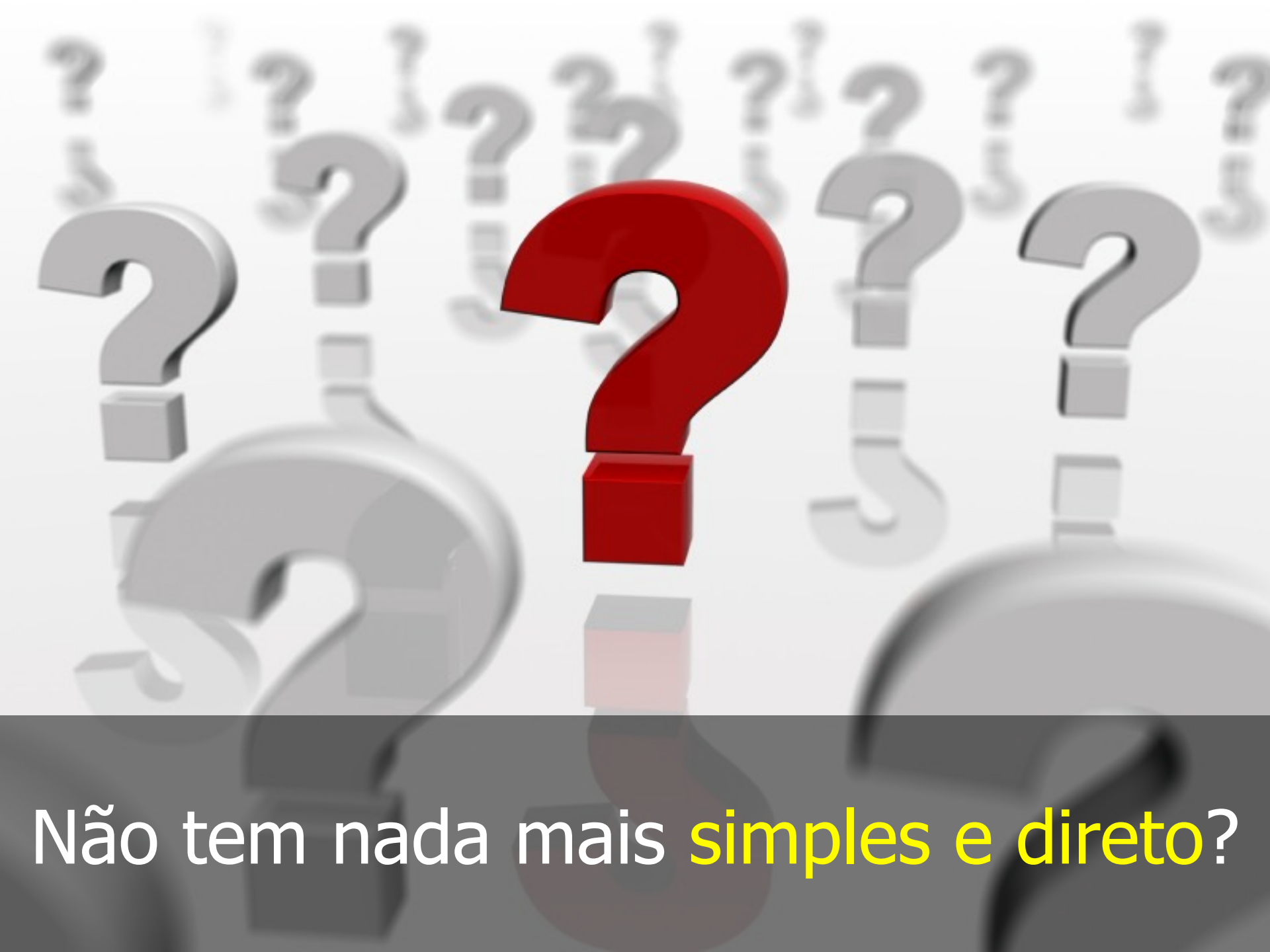
clearTimeout.js

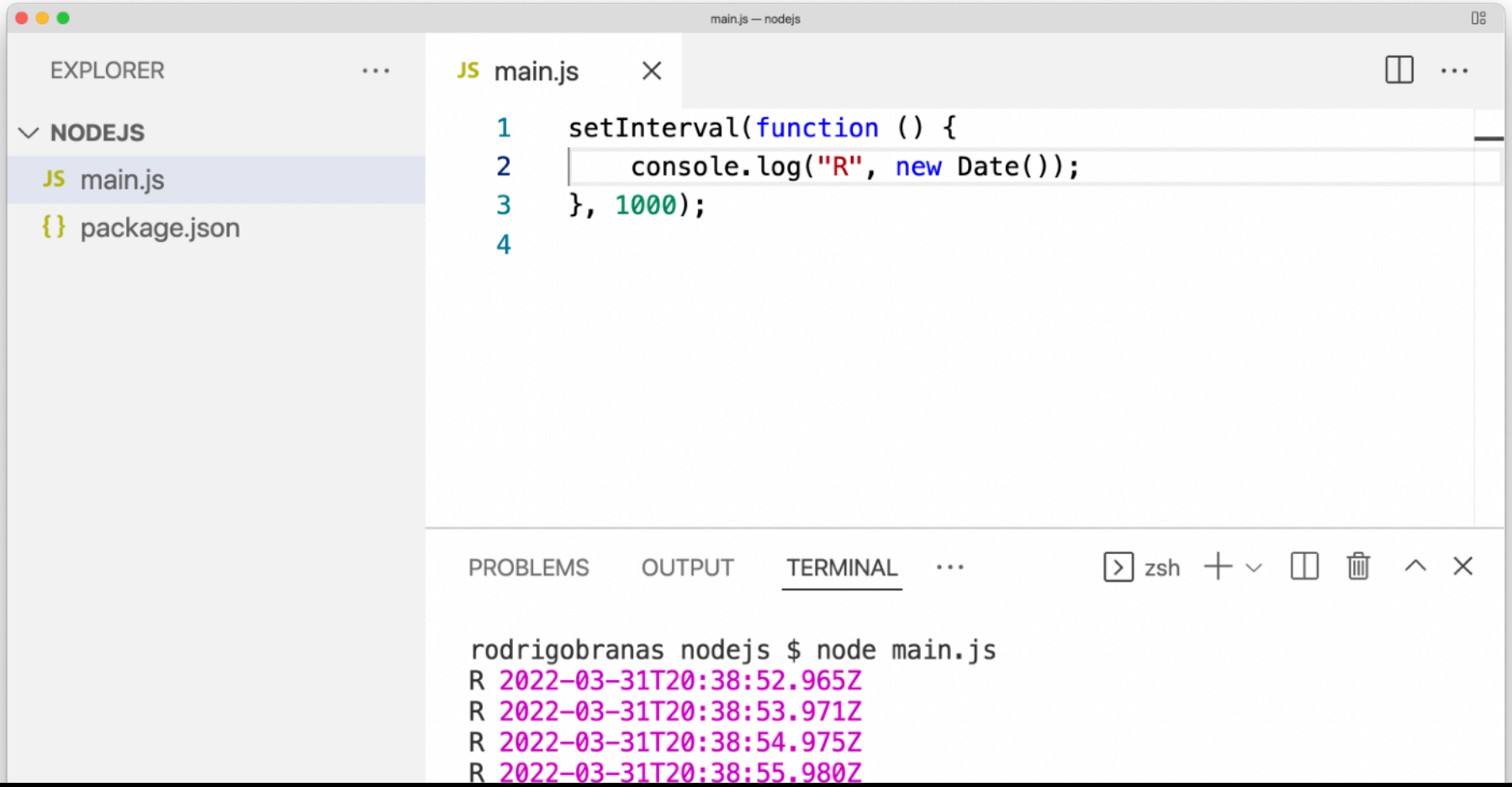


É possível também criar um timer que execute a cada período de tempo

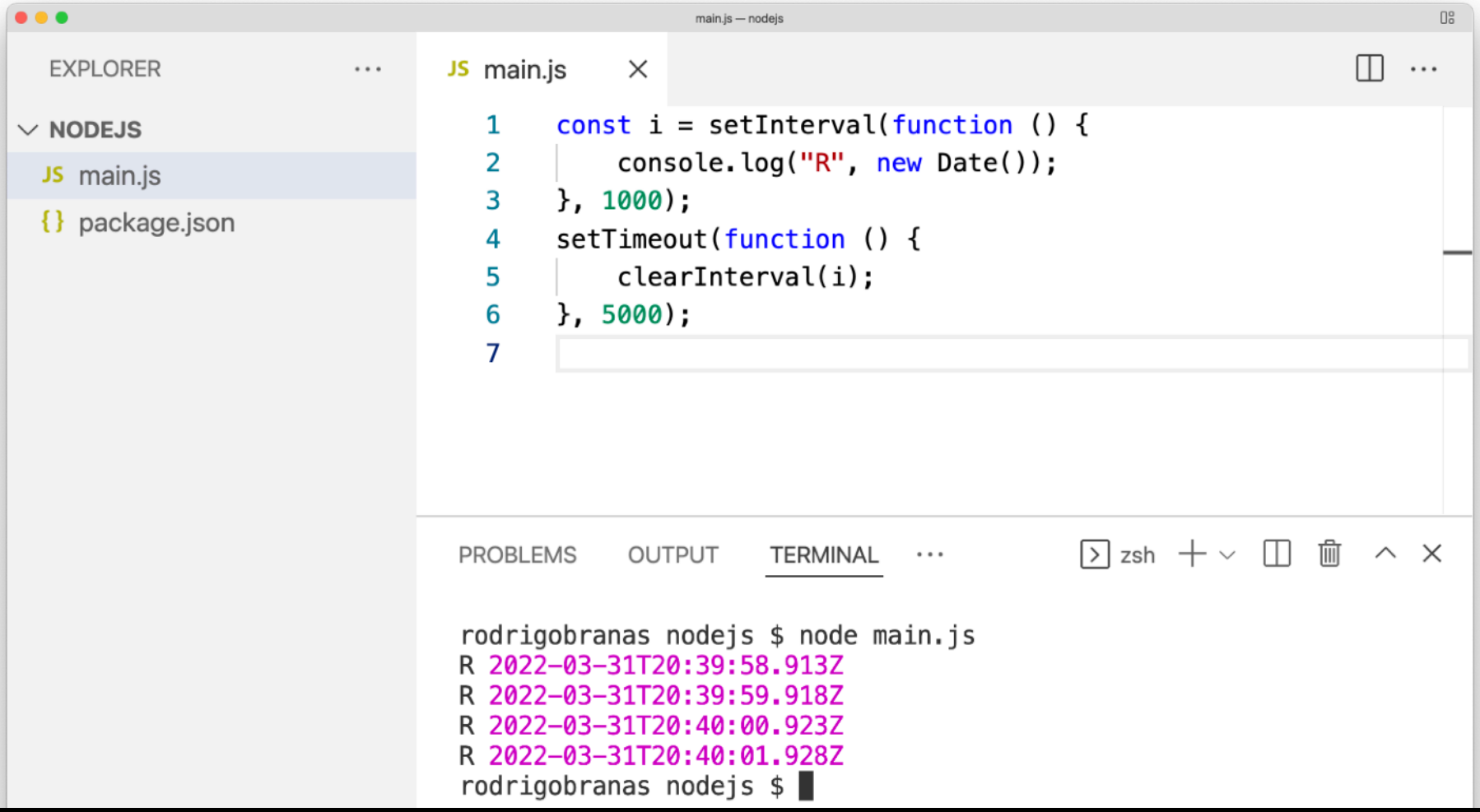


Criando um timer com setTimeout


Não tem nada mais simples e direto?

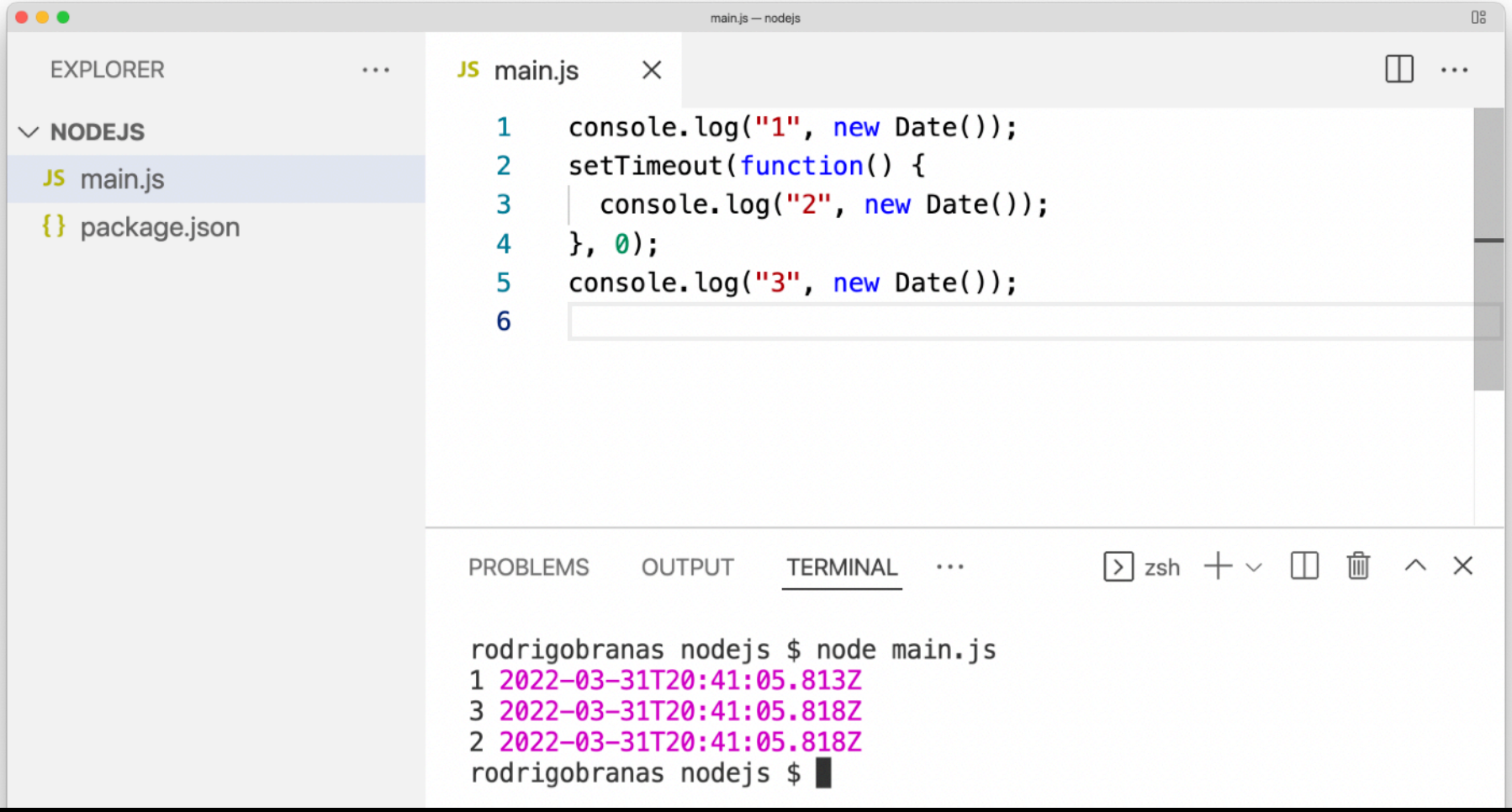


Criando um timer com setInterval

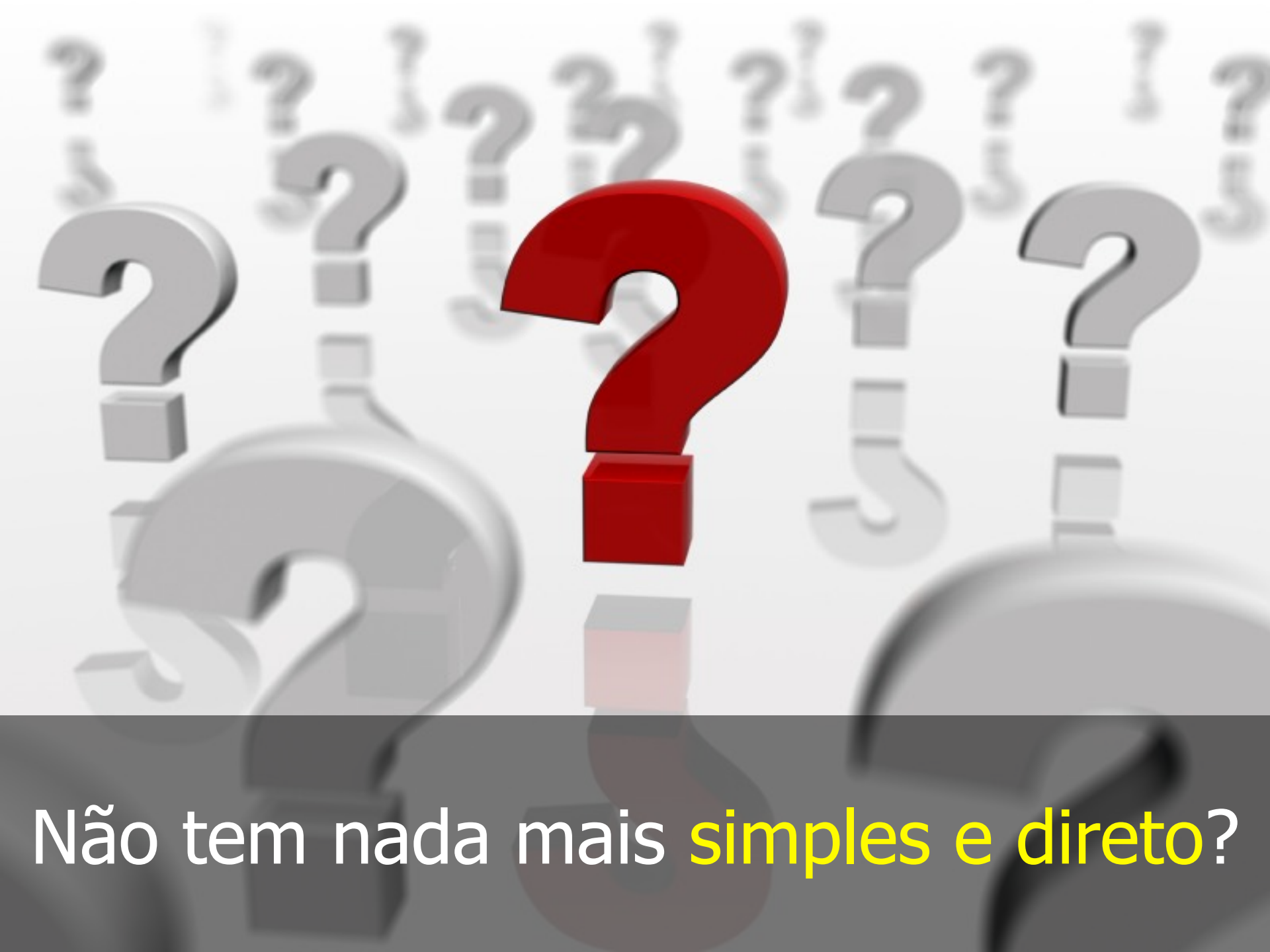


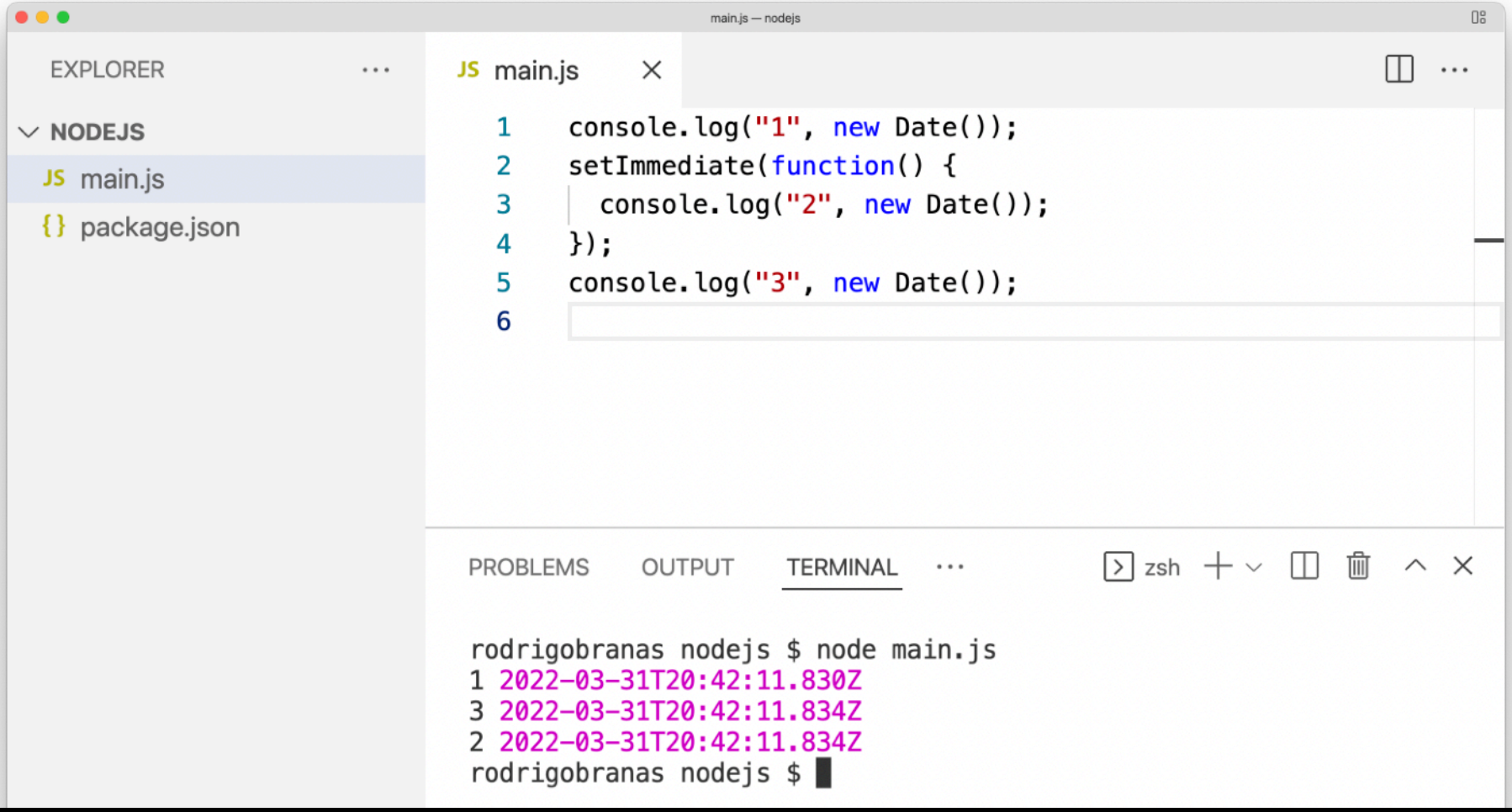
Cancelando um timer com clearInterval

E se a intenção for apenas colocar a função na fila, para executar depois?



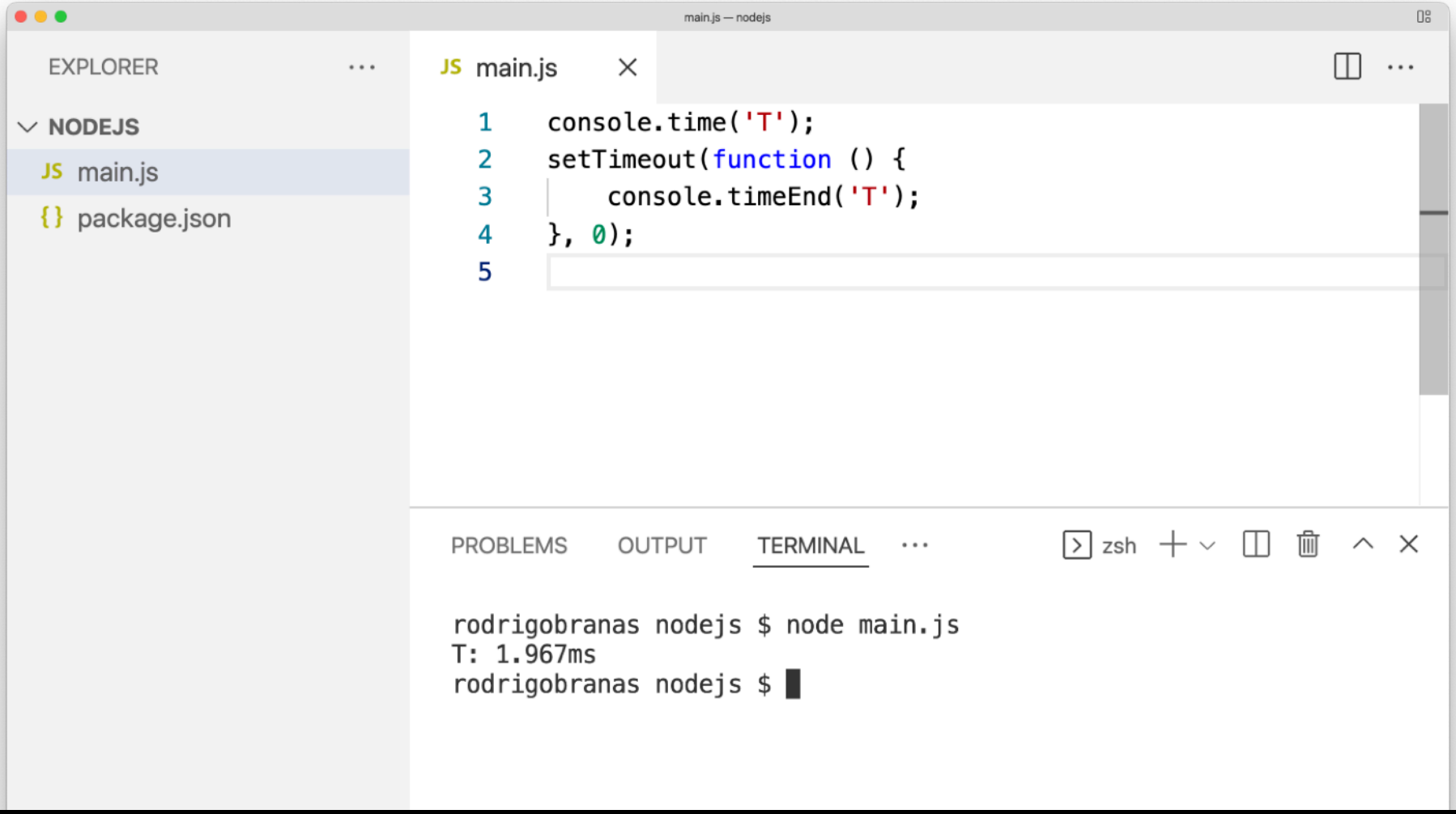
Criando um timer imediato


Não tem nada mais simples e direto?

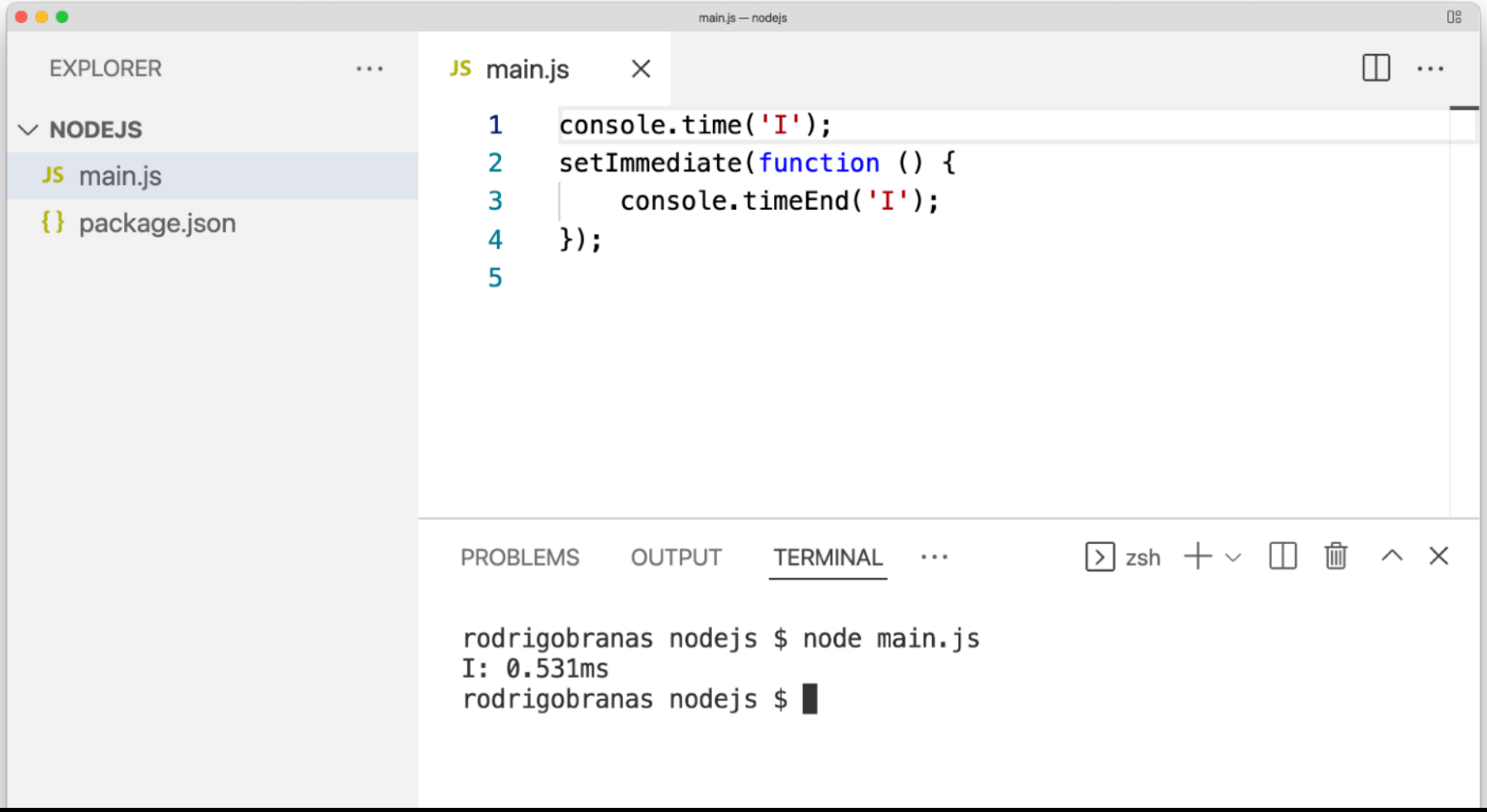


Criando um timer imediato

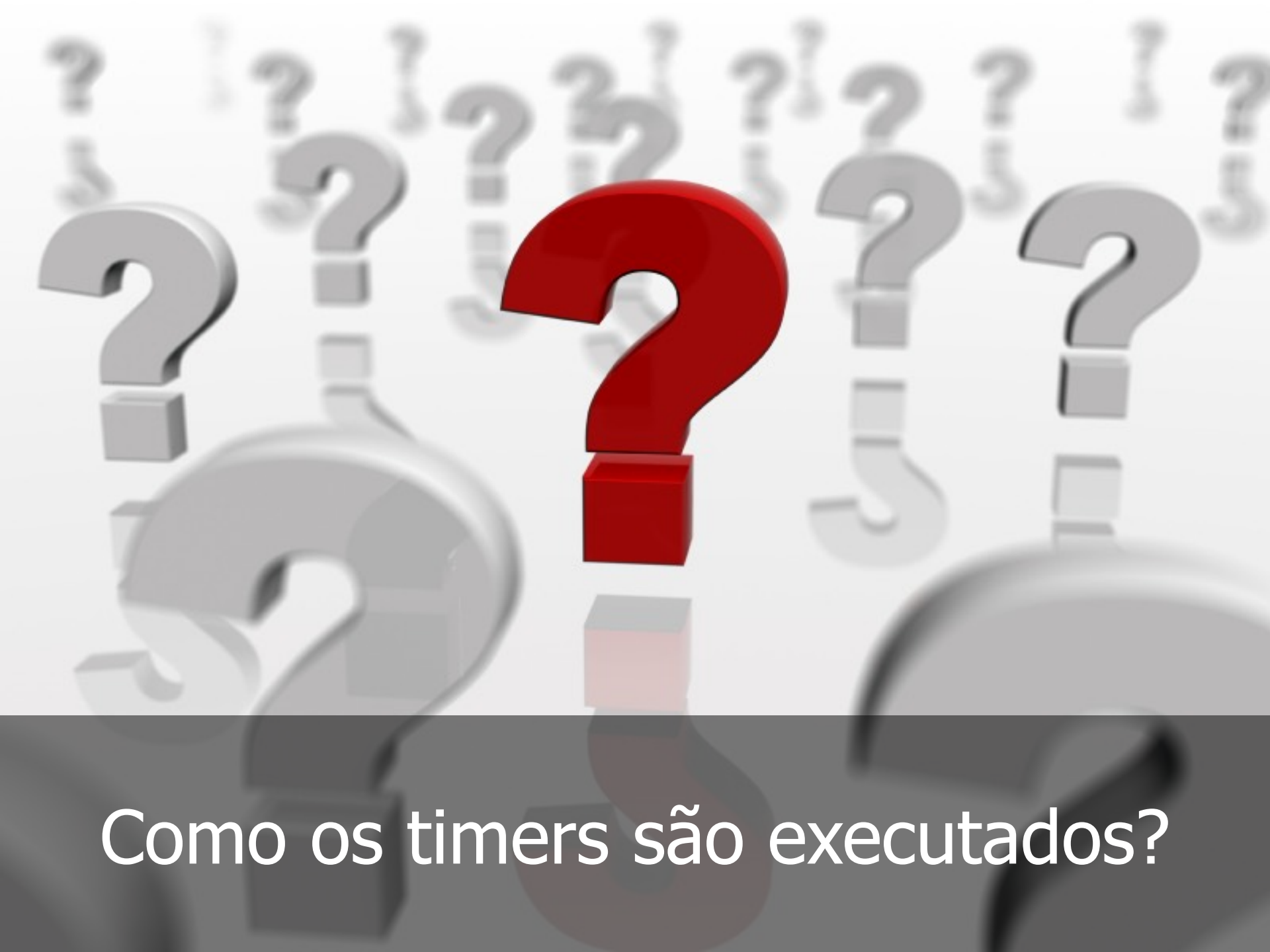
Existe diferença de performance entre o setTimeout e o setImmediate?



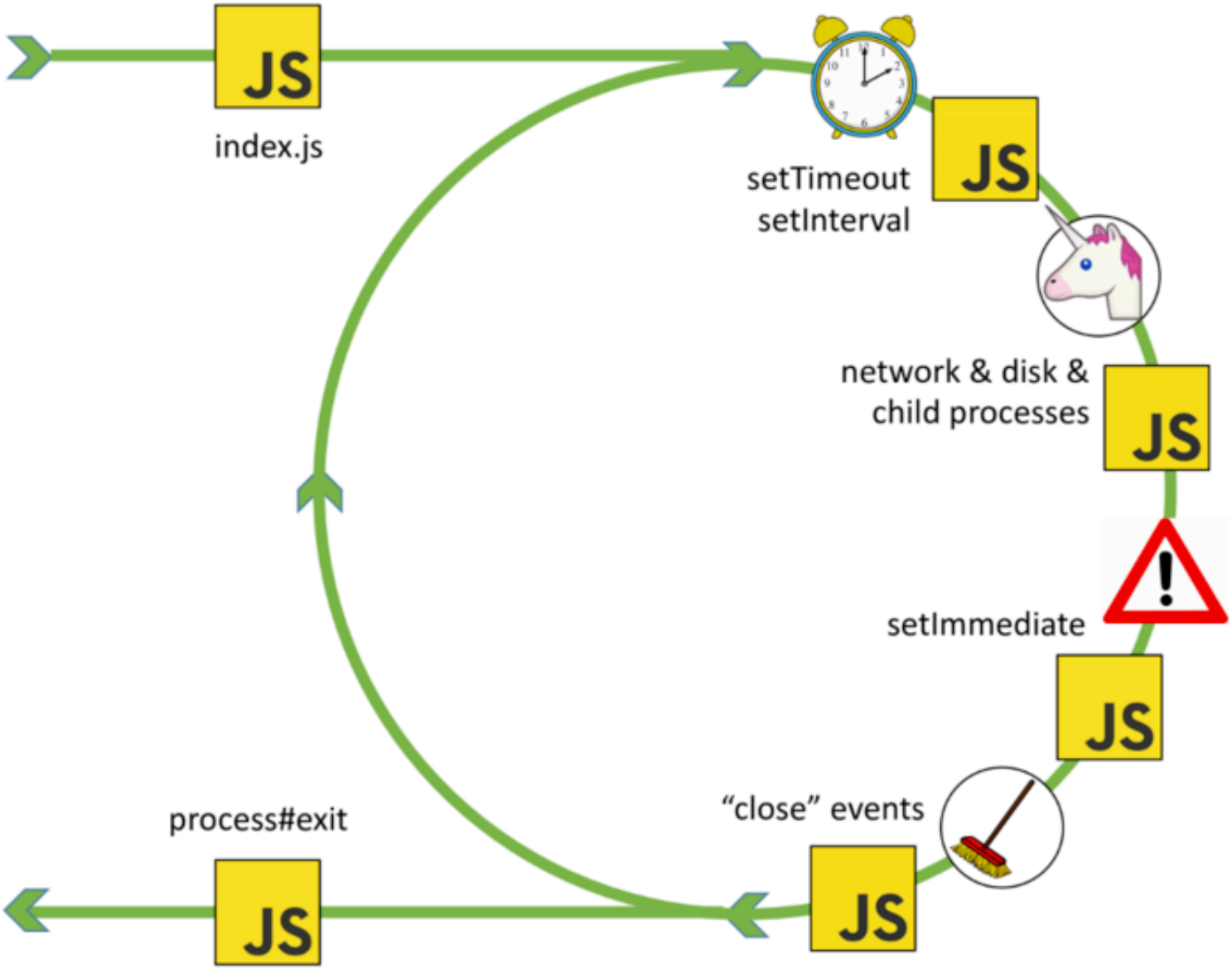
Medindo a performance do setTimeout

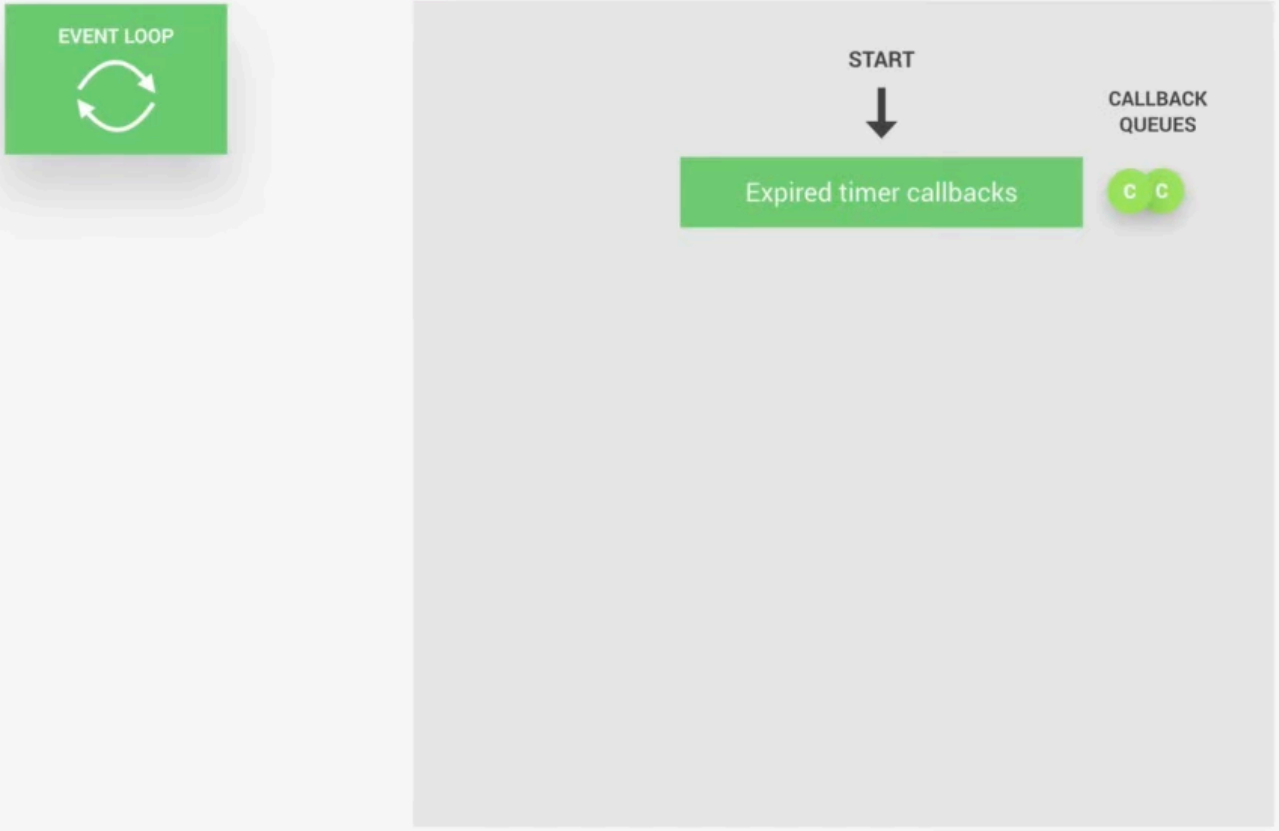


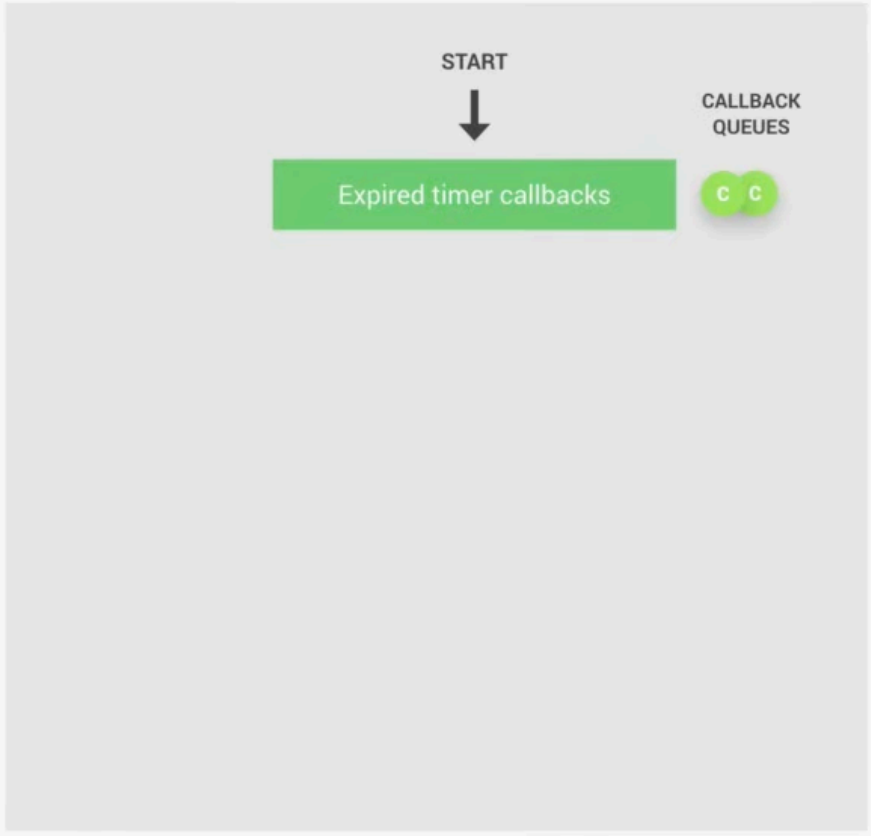
Medindo a performance do setImmediate



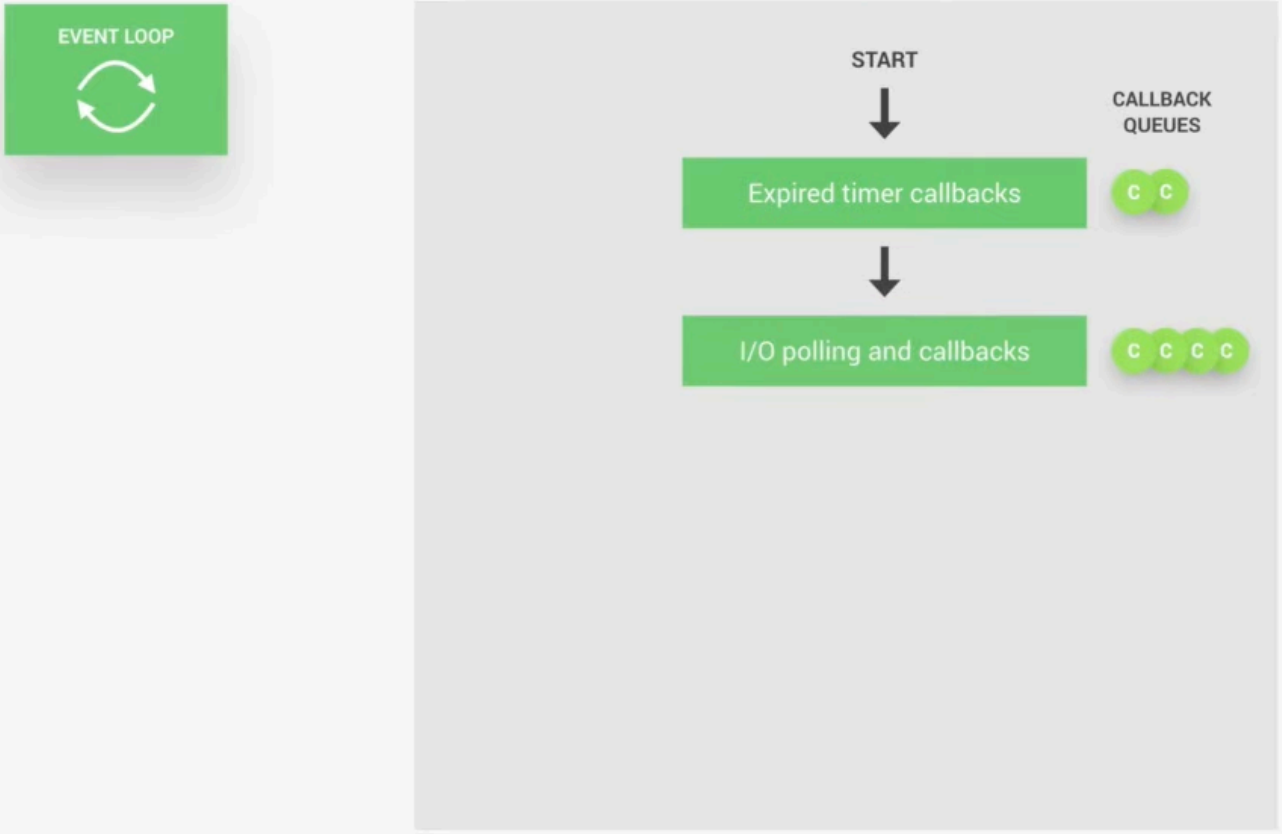
Como os timers são executados?


JS
index.js
setTimeout
setInterval
JS
network & disk &
child processes
JS
setImmediate
JS
"close" events
JS
process#exit
JS

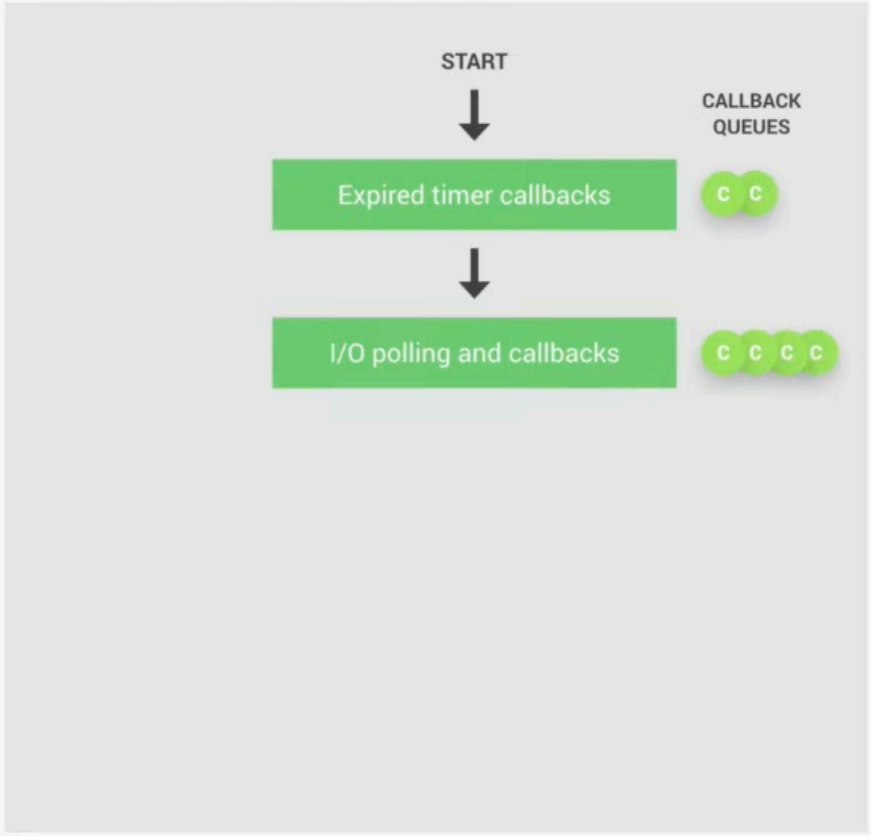

EVENT LOOP
START
CALLBACK QUEUES
Expired timer callbacks
C
C



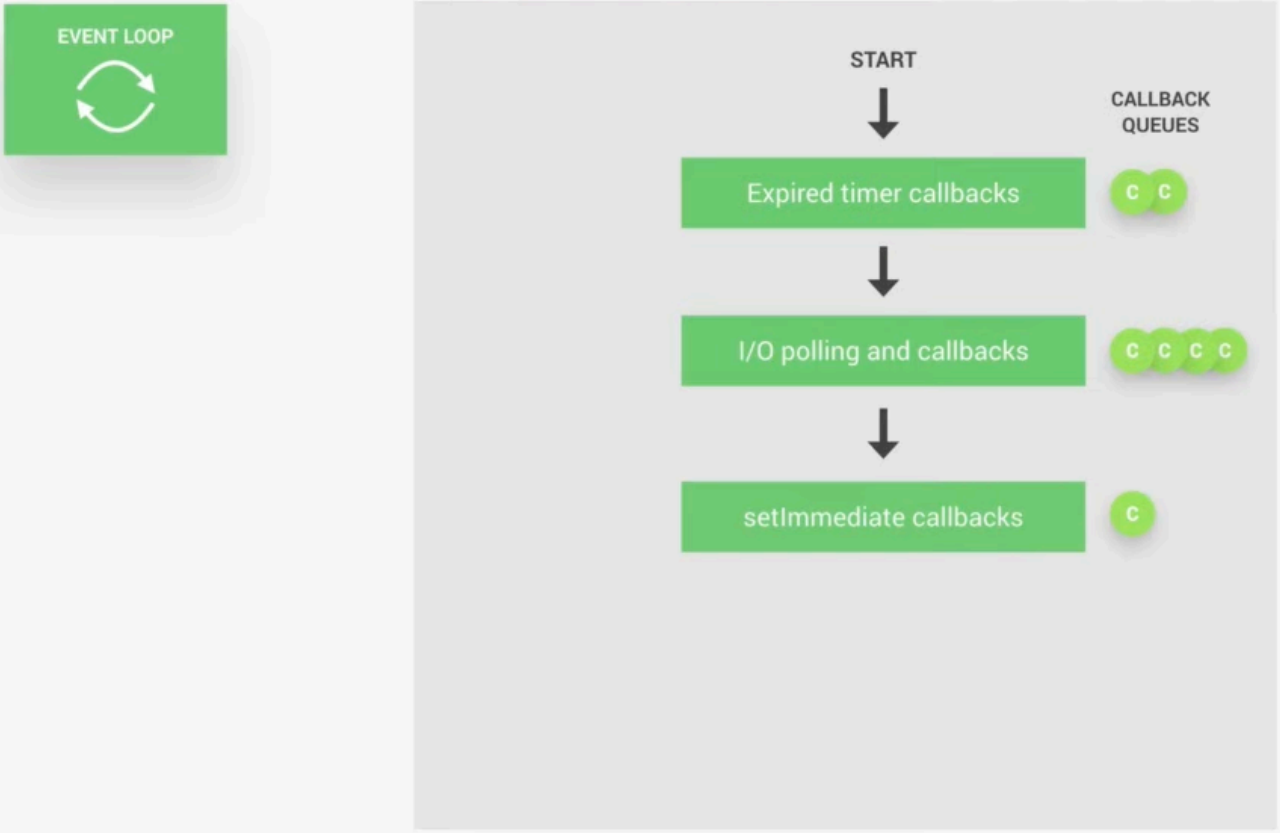

```
setTimeout(() => {
    console.log('Timer expired!');
});
```

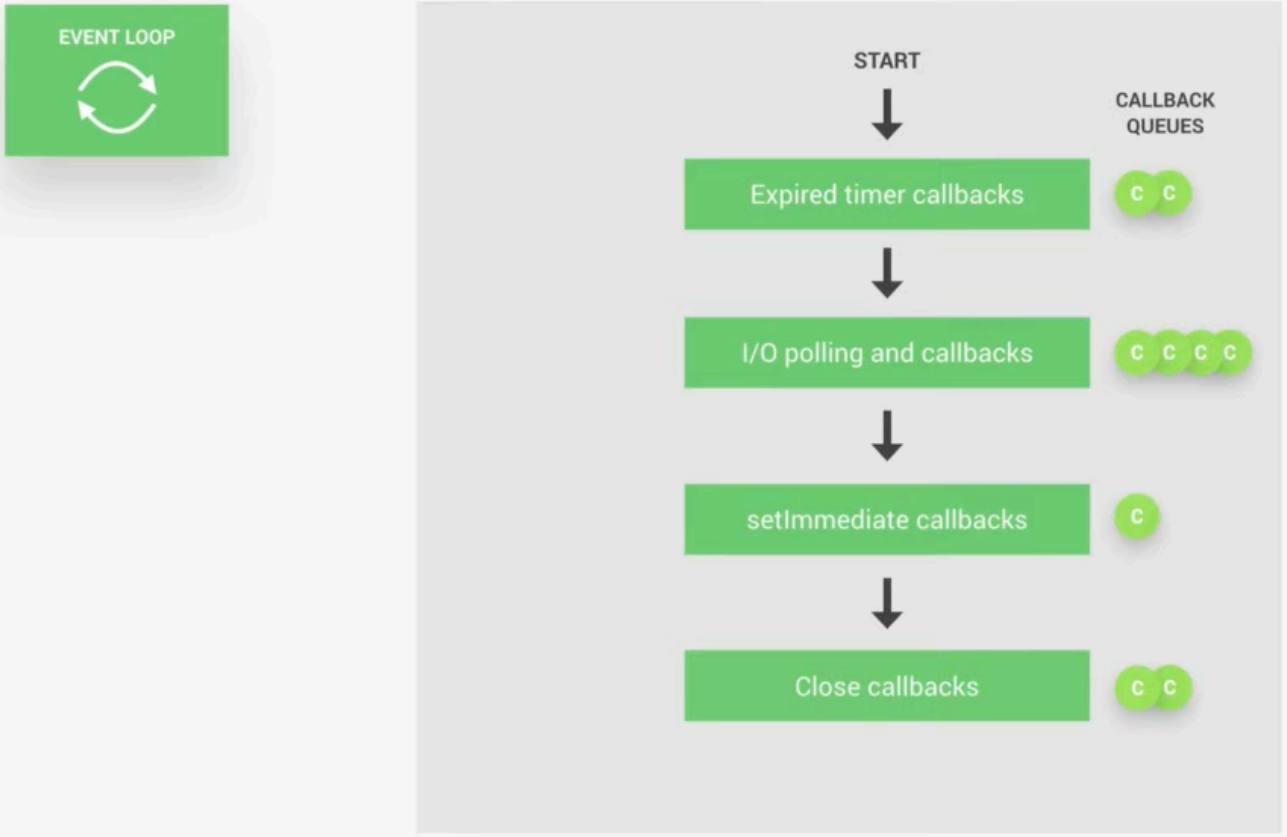

EVENT LOOP
START
CALLBACK QUEUES
Expired timer callbacks
C C
I/O polling and callbacks
C C C C

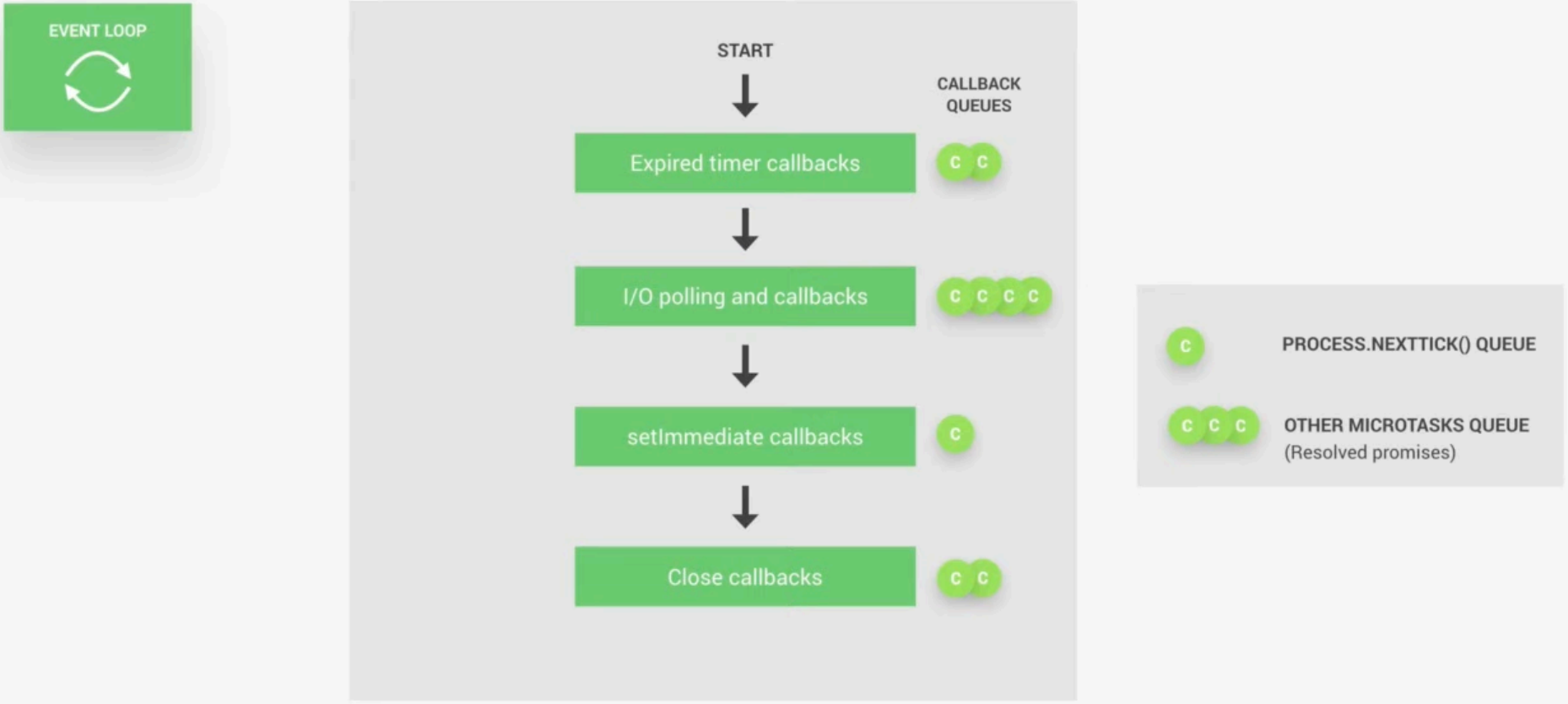
EVENT LOOP



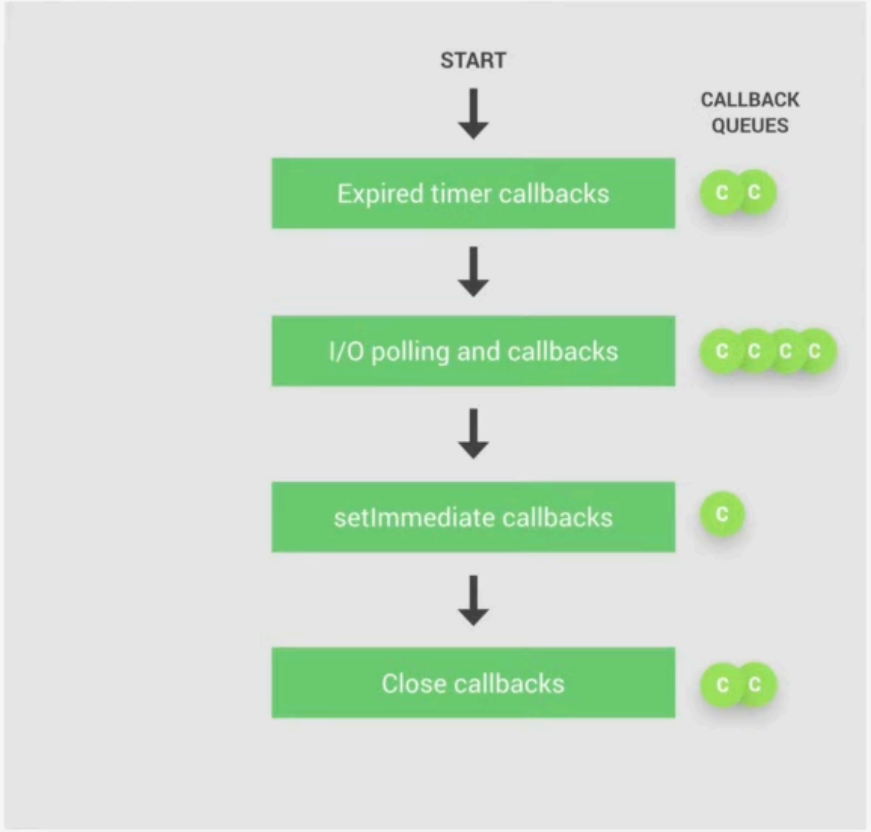

```
fs.readFile('file.txt', (e, d) => {
    console.log('File read!');
});
```


EVENT LOOP
START
CALLBACK QUEUES
Expired timer callbacks
I/O polling and callbacks
setImmediate callbacks


EVENT LOOP
START
CALLBACK QUEUES
Expired timer callbacks
I/O polling and callbacks
setImmediate callbacks
Close callbacks


EVENT LOOP
START
CALLBACK QUEUES
Expired timer callbacks
I/O polling and callbacks
setImmediate callbacks
Close callbacks
PROCESS.NEXTTICK() QUEUE
OTHER MICROTASKS QUEUE
(Resolved promises)

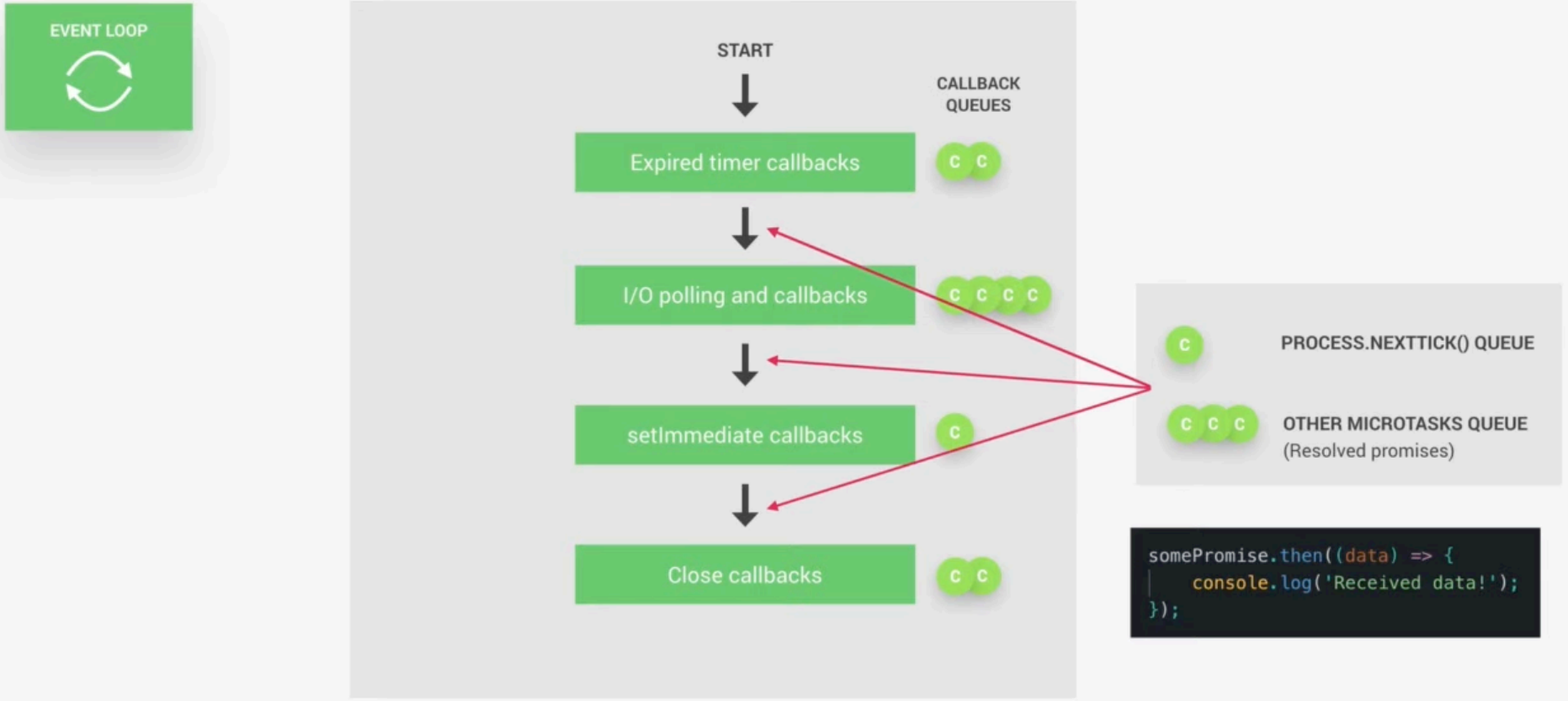
EVENT LOOP

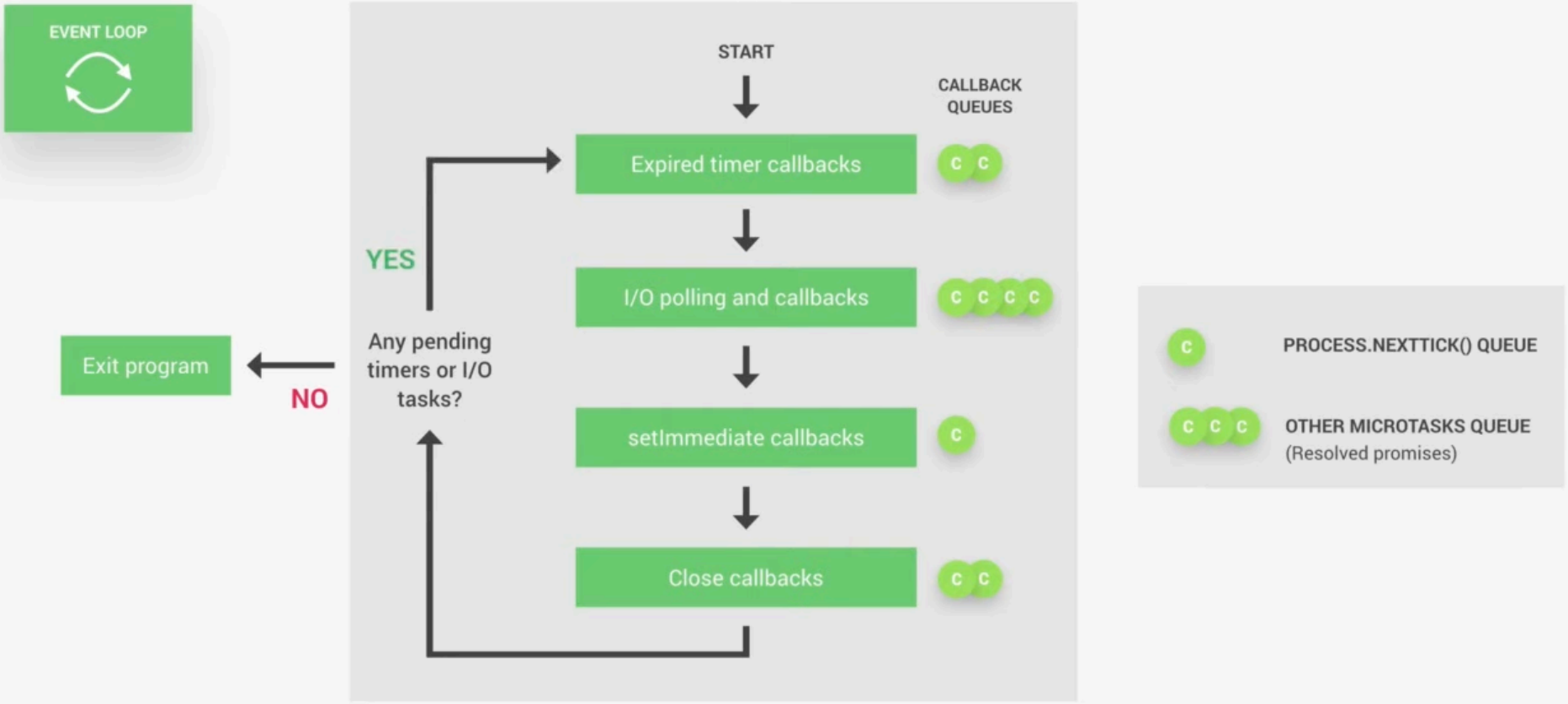


PROCESS.NEXTTICK() QUEUE

OTHER MICROTASKS QUEUE
(Resolved promises)

```
somePromise.then((data) => {
    console.log('Received data!');
});
```


EVENT LOOP
START
CALLBACK QUEUES
Expired timer callbacks
I/O polling and callbacks
setImmediate callbacks
Close callbacks
PROCESS.NEXTTICK() QUEUE
OTHER MICROTASKS QUEUE
(Resolved promises)
somePromise.then((data) => {
console.log('Received data!');
});


EVENT LOOP
START
CALLBACK QUEUES
Expired timer callbacks
I/O polling and callbacks
setImmediate callbacks
Close callbacks
YES
Any pending timers or I/O tasks?
NO
Exit program
PROCESS.NEXTTICK() QUEUE
OTHER MICROTASKS QUEUE
(Resolved promises)

Qual é a diferença entre o nextTick e o setImmediate?