COMMAND LINE. DEBUGGING.

Agenda

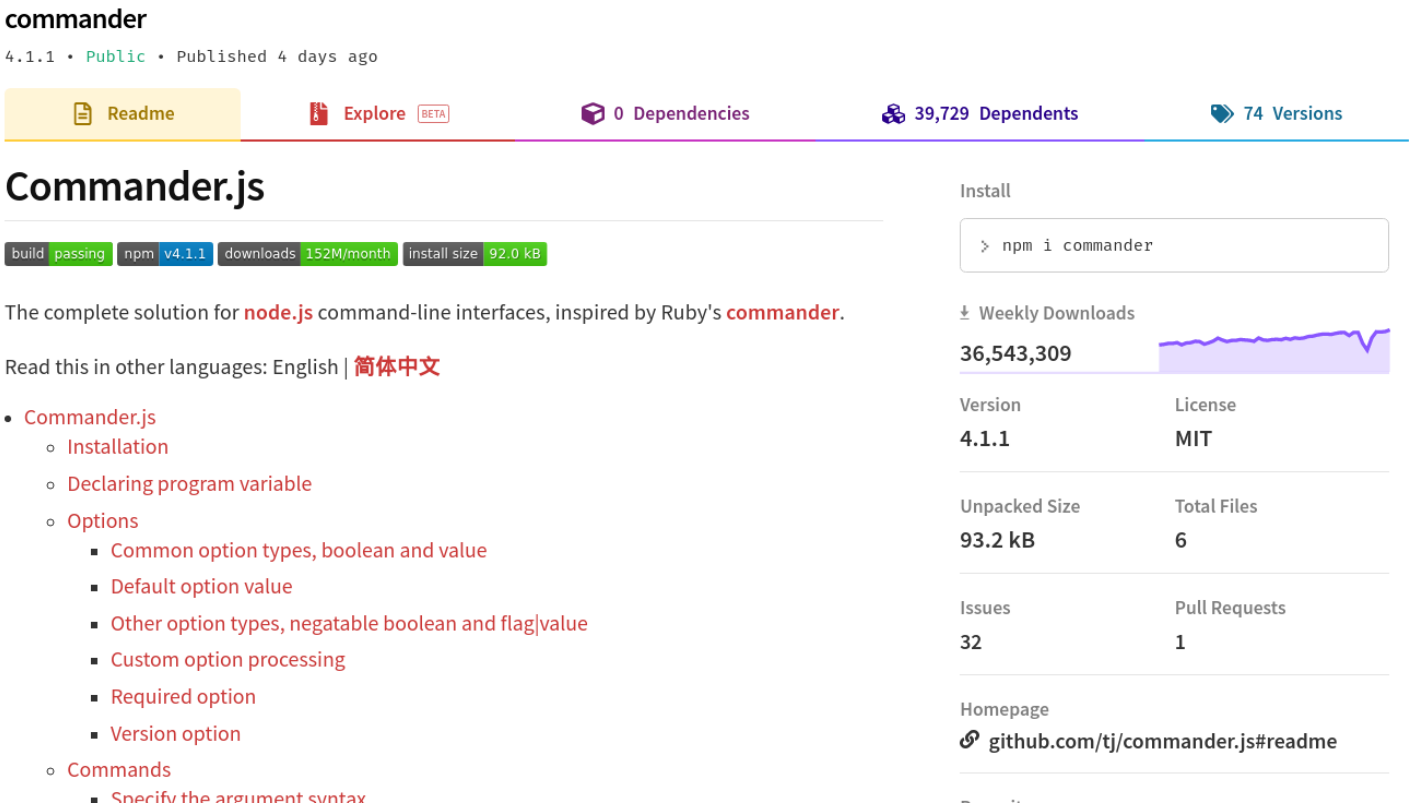
- Command-line arguments
- Parsing command-line arguments with `commander` package
- Enable debug output for build-in packages
- `debug` package
- Debugging NodeJS with Chrome

COMMAND LINE ARGUMENTS

Command-line arguments

```
console.log(process.argv);

/**
 $ node index.js one two=three four
 [
   '/path/to/executable/node', // process.execPath
   '/path/to/script/index.js',
   'one',
   'two=three',
   'four'
 ]
 */
```

Parsing command-line arguments with `commander` package



commander

4.1.1 • Public • Published 4 days ago

Readme | Explore BETA | 0 Dependencies | 39,729 Dependents | 74 Versions

Commander.js

build passing | npm v4.1.1 | downloads 152M/month | install size 92.0 kB

The complete solution for **node.js** command-line interfaces, inspired by Ruby's **commander**.

Read this in other languages: English | **简体中文**

- Commander.js
  - Installation
  - Declaring program variable
  - Options
    - Common option types, boolean and value
    - Default option value
    - Other option types, negatable boolean and flag|value
    - Custom option processing
    - Required option
    - Version option
  - Commands
    - Specify the argument syntax

Install

```
> npm i commander
```

Weekly Downloads

36,543,309

Version	License
4.1.1	MIT

Unpacked Size	Total Files
93.2 kB	6

Issues	Pull Requests
32	1

Homepage

github.com/tj/commander.js#readme

Parsing command-line arguments with `commander` package

Key features:

- command line options, including aliases (e.g. `script.js --option` = `command -o`)
- optional, mandatory options, default values, options post-processing
- sub commands (e.g. `script.js subcommand`)
- automatic `--help` option generation

Parsing command-line arguments with `commander` package

```
const program = require('commander');

program
  .option( flags: '-b, --bool', description: 'boolean value')
  .option( flags: '-s, --string <value>', description: 'string value')
  .option( flags: '-d, --default <value>', description: 'string with default value', defaultValue: 'default')
  .option( flags: '-B, --no-negative-bool', description: 'boolean with false when used')
  .option( flags: '-i, --int <value>', description: 'integer value', fn: v => parseInt(v, radix: 10), defaultValue: 0)
  .option( flags: '-m, --multi <value>', description: 'array of values', fn: (val, prev) => [...prev, val], defaultValue: [])
  .parse(process.argv);

console.log(`Bool:          ${program.bool}`);
console.log(`String:        ${program.string}`);
console.log(`Default:       ${program.default}`);
console.log(`Negative bool: ${program.negativeBool}`);
console.log(`Int:           ${program.int}`);
console.log(`Multi:         ${program.multi}`);
```

Parsing command-line arguments with `commander` package

```
$ node index.js --bool --string hello --no-negative-bool --int r42 --multi 1 --multi 2 --multi 3
Bool:          true
String:        hello
Default:       default
Negative bool: false
Int:           NaN
Multi:         1,2,3
```

```
$ node index.js -b -s hello -B -i 42 -m 1 -m 2 -m 3
Bool:          true
String:        hello
Default:       default
Negative bool: false
Int:           42
Multi:         1,2,3
```

Parsing command-line arguments with `commander` package

```
$ node index.js --help
Usage: index [options]

Options:
  -b, --bool                boolean value
  -s, --string <value>      string value
  -d, --default <value>     string with default value (default: "default")
  -B, --no-negative-bool    boolean with false when used
  -i, --int <value>         integer value (default: 0)
  -m, --multi <value>       array of values (default: [])
  -h, --help                output usage information
```

Parsing command-line arguments with `commander` package

```
const program = require('commander');

const int = v => parseInt(v, radix: 10);

program
  .command( nameAndArgs: 'new <migration-name>')
  .description( str: 'creates a new migration')
  .action( fn: name => {
    console.log(`New migration ${name} created`)
  });

program
  .command( nameAndArgs: 'run')
  .description( str: 'applies migration(s)')
  .option( flags: '-c, --count <v>', description: 'number of migration to run', int, defaultValue: 1)
  .action( fn: args => {
    console.log(`Applying ${args.count} migration`)
  });


program.parse(process.argv);
```

Parsing command-line arguments with `commander` package

```
$ node index.js --help
Usage: index [options] [command]

Options:
  -h, --help             output usage information

Commands:
  new <migration-name>  creates a new migration
  run [options]         applies migration(s)
```

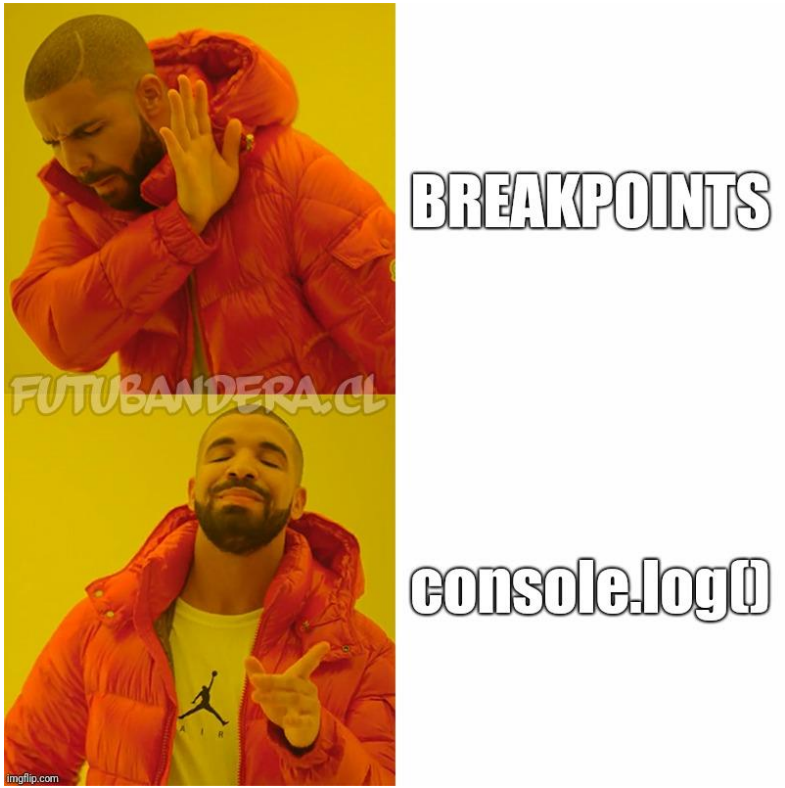
```
$ node index.js run --help
Usage: index run [options]

applies migration(s)

Options:
  -c, --count <v>  number of migration to run (default: 1)
  -h, --help       output usage information
```

DEBUGGING

Debugging

Debugging is the process of finding and resolving defects or problems within a computer program that prevent correct operation of computer software or a system.

Wikipedia



Enable debug output for build-in packages

```
// $ NODE_DEBUG=http node index.js

const {request} = require('http');

const req = request('http://httpbin.org/json', {
  method: 'GET'
});

req.on( event: 'response', listener: async res => {
  const chunks = [];
  for await (const chunk of res) {
    chunks.push(chunk)
  }

  const body = JSON.parse(Buffer.concat(chunks).toString());

  console.dir(body, options: {depth: null});
});

req.end();
```

`debug` package

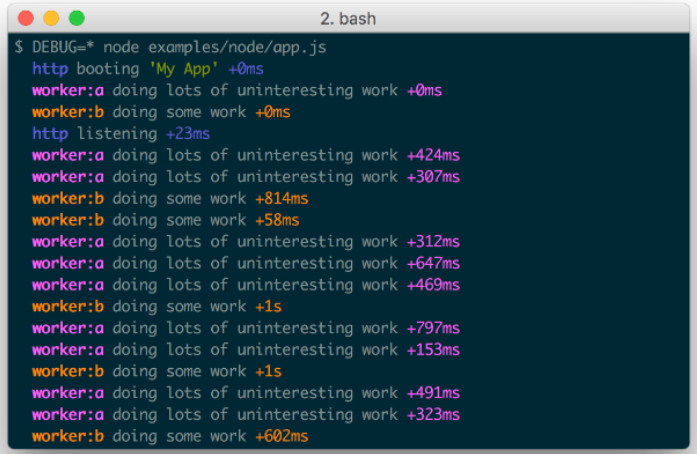
debug

4.1.1 • Public • Published a year ago

Readme | Explore BETA | 1 Dependency | 32,787 Dependents | 64 Versions

debug

build passing | coverage 88% | Slack | backers 11 | sponsors 6



Install

```
> npm i debug
```

Weekly Downloads

57,914,995

Version	License
4.1.1	MIT

Unpacked Size	Total Files
81.5 kB	9

Issues	Pull Requests
24	15

Homepage

github.com/visionmedia/debug#readme

`debug` package

```
1  // $ DEBUG=http node index.js
2
3  const app = require('express')();
4  const port = 3000;
5
6  app.use( fn: (req, res) ⇒ res.end('Hello world!'));
7
8  app.listen(port);
```

`debug` package

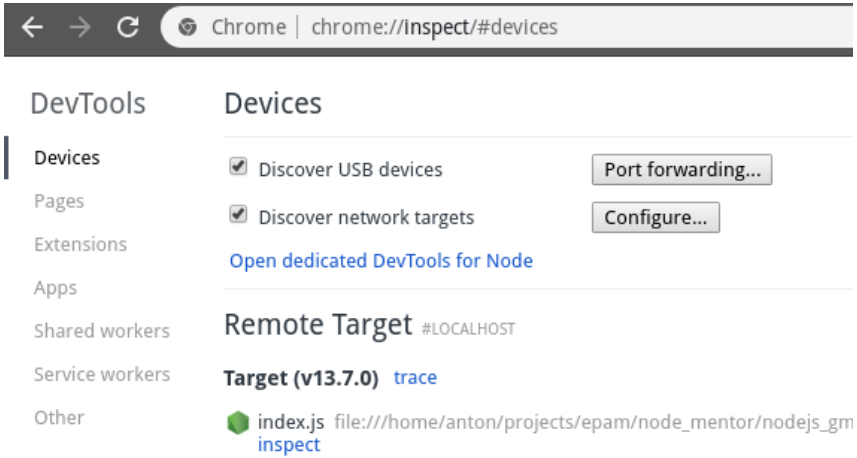
```
DEBUG=express:* node index.js
express:application set "x-powered-by" to true +0ms
express:application set "etag" to 'weak' +2ms
express:application set "etag fn" to [Function: generateETag] +0ms
express:application set "env" to 'development' +1ms
express:application set "query parser" to 'extended' +0ms
express:application set "query parser fn" to [Function: parseExtendedQueryString] +0ms
express:application set "subdomain offset" to 2 +0ms
express:application set "trust proxy" to false +1ms
express:application set "trust proxy fn" to [Function: trustNone] +0ms
express:application booting in development mode +0ms
express:application set "view" to [Function: View] +0ms
express:application set "views" to '/home/anton/projects/epam/node_mentor/nodejs_gmp_20
express:application set "jsonp callback name" to 'callback' +1ms
express:router use '/' query +0ms
express:router:layer new '/' +0ms
express:router use '/' expressInit +1ms
express:router:layer new '/' +0ms
express:router use '/' <anonymous> +0ms
express:router:layer new '/' +0ms
```

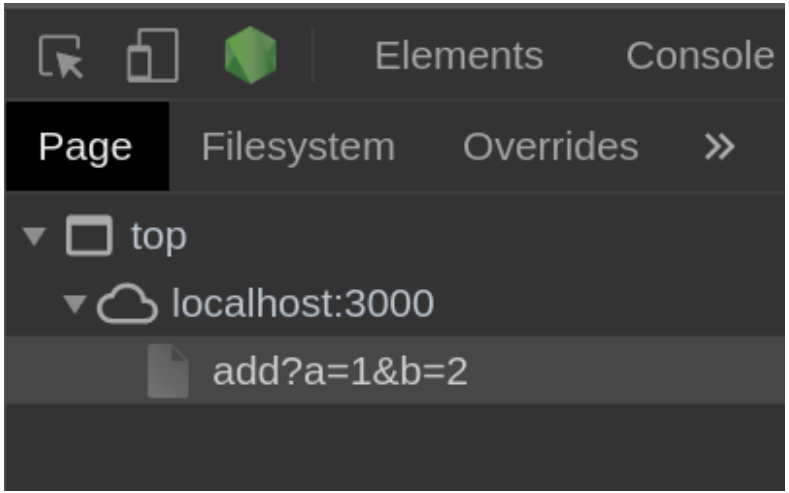
Debugging NodeJS with Chrome

```
const app = require('express')();
const port = 3000;

// GET /add?a=1&b=2 ⇒ 3
app.get('/add', (req, res, next) ⇒ {
  const {a, b} = req.query;
  res.send(a + b)
});

app.listen(port);
```

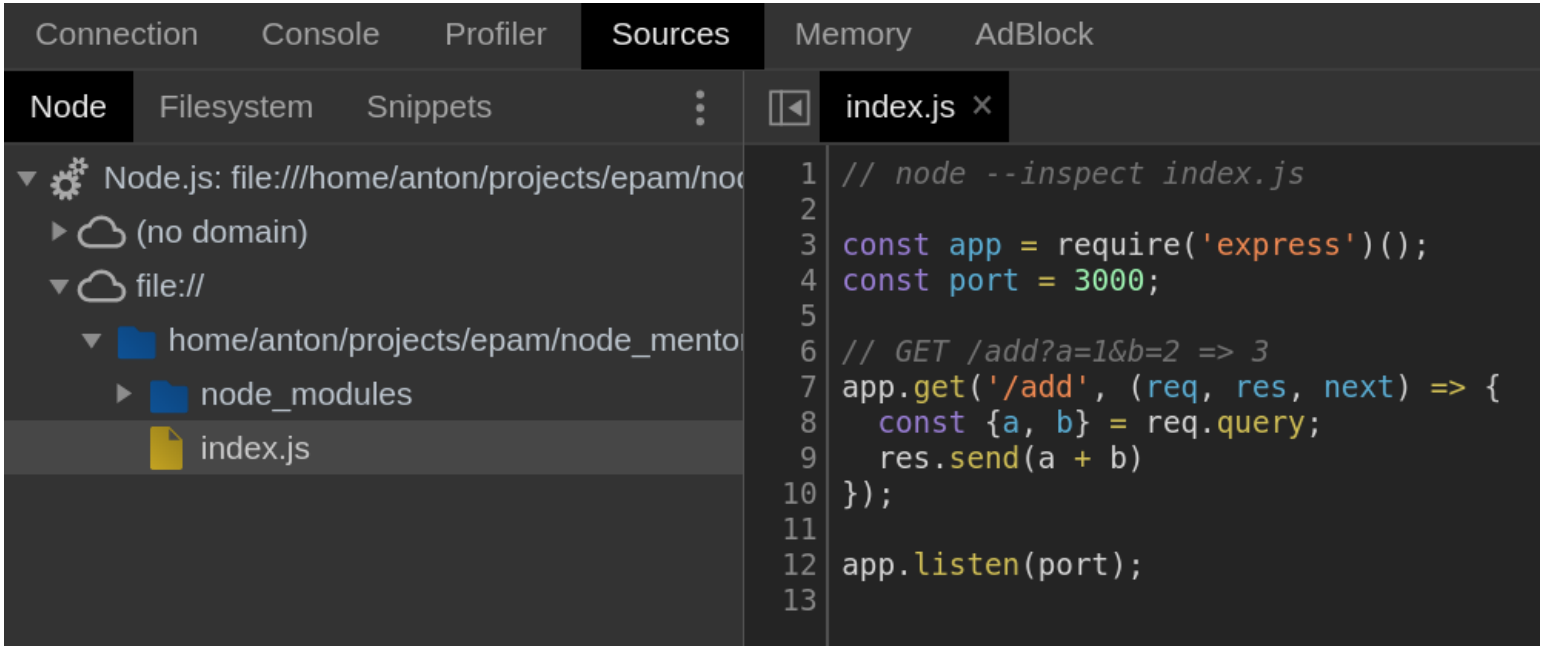
```
$ node --inspect index.js
Debugger listening on ws://127.0.0.1:9229/6f47828f-aee6-4766-8aa2-7414072fbb90
For help, see: https://nodejs.org/en/docs/inspector
```

Debugging NodeJS with Chrome

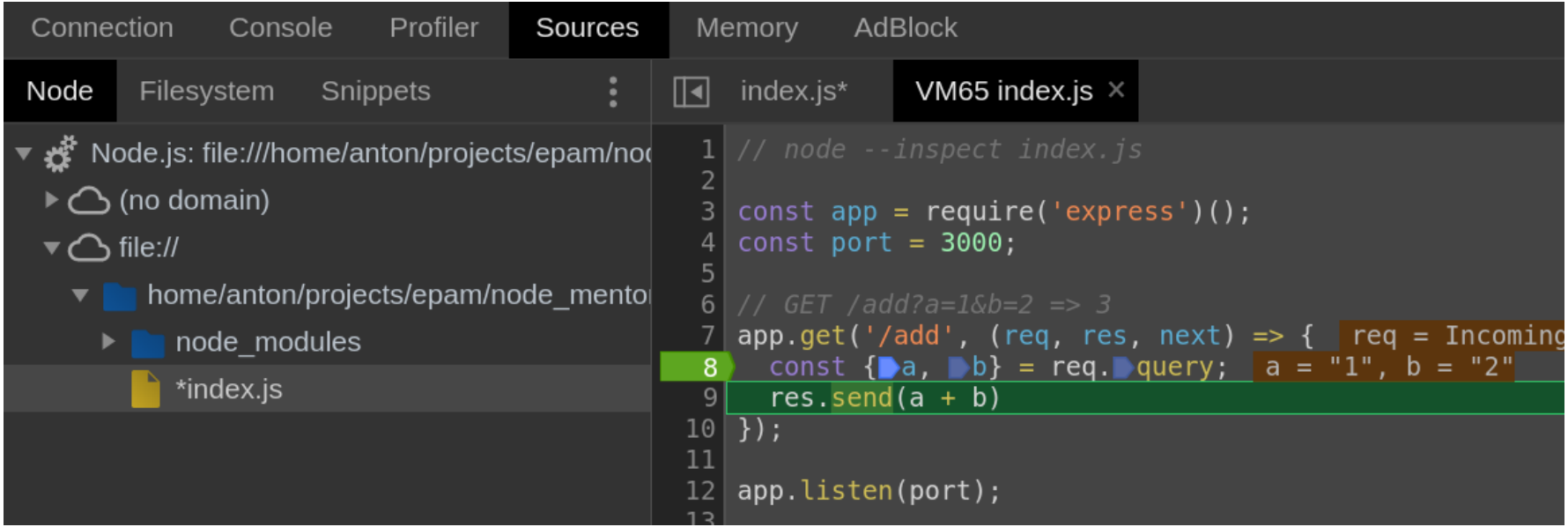




Debugging NodeJS with Chrome



Debugging NodeJS with Chrome



Debugging NodeJS with Chrome

node --inspect=[host:port] *script.js*	Bind to address or hostname *host* (default: 127.0.0.1) Listen on *ort* (default: 9229)
node --inspect-brk=[host:port] *script.js*	Bind to address or hostname *host* (default: 127.0.0.1) Listen on *port* (default: 9229) Break before user code starts
node inspect *script.js*	Spawn child process to run user's script under --inspect flag; and use main process to run CLI debugger.
node inspect --port=[*port*]	Spawn child process to run user's script under --inspect flag; and use main process to run CLI debugger. Listen on *port* (default: 9229)

Useful links

- Source code
- commander package
- debug package
- Debugging Guide

Q & A