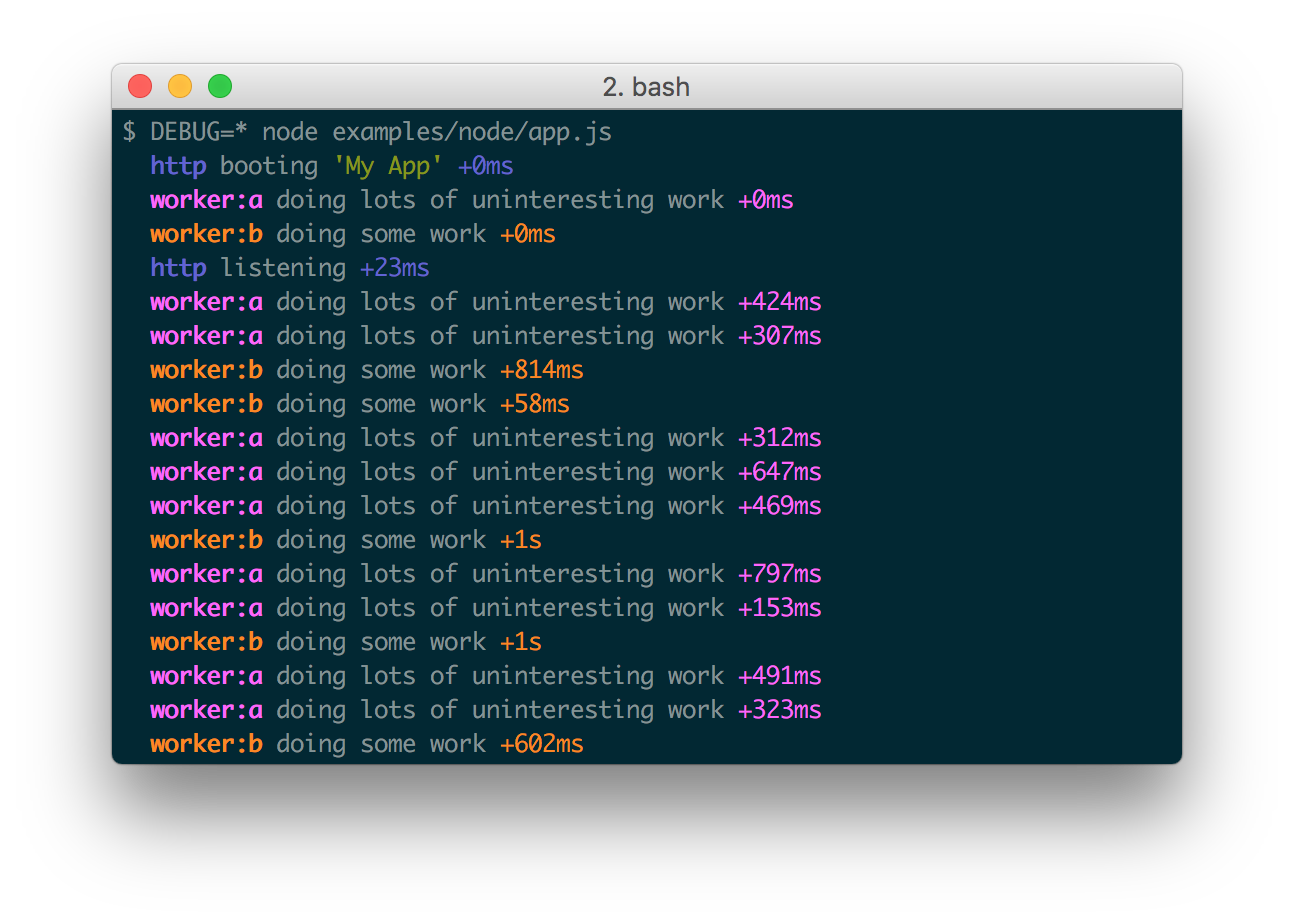
# debug

[Build Status](https://travis-ci.org/debug-js/debug) [Coverage Status](https://coveralls.io/github/debug-js/debug?branch=master) [Slack](https://visionmedia-community-slackin.now.sh/) [OpenCollective](#2bn6wsx) [OpenCollective](#qsh70q)

 A tiny JavaScript debugging utility modelled after Node.js core's debugging technique. Works in Node.js and web browsers.

## Installation

$ npm install debug

## Usage

debug exposes a function; simply pass this function the name of your module, and it will return a decorated version of console.error for you to pass debug statements to. This will allow you to toggle the debug output for different parts of your module as well as the module as a whole.

Example [*app.js*](http://./examples/node/app.js):

var debug = require('debug')('http')

, http = require('http')

, name = 'My App';

// fake app

debug('booting %o', name);

http.createServer(function(req, res){

debug(req.method + ' ' + req.url);

res.end('hello\n');

}).listen(3000, function(){

debug('listening');

});

// fake worker of some kind

require('./worker');

Example [*worker.js*](http://./examples/node/worker.js):

var a = require('debug')('worker:a')

, b = require('debug')('worker:b');

function work() {

a('doing lots of uninteresting work');

setTimeout(work, Math.random() \* 1000);

}

work();

function workb() {

b('doing some work');

setTimeout(workb, Math.random() \* 2000);

}

workb();

The DEBUG environment variable is then used to enable these based on space or comma-delimited names.

Here are some examples:

#### screen shot 2017-08-08 at 12 53 04 pm screen shot 2017-08-08 at 12 53 38 pm screen shot 2017-08-08 at 12 53 25 pm Windows command prompt notes

##### CMD

On Windows the environment variable is set using the set command.

set DEBUG=\*,-not\_this

Example:

set DEBUG=\* & node app.js

##### PowerShell (VS Code default)

PowerShell uses different syntax to set environment variables.

$env:DEBUG = "\*,-not\_this"

Example:

$env:DEBUG='app';node app.js

Then, run the program to be debugged as usual.

npm script example:

"windowsDebug": "@powershell -Command $env:DEBUG='\*';node app.js",

## Namespace Colors

Every debug instance has a color generated for it based on its namespace name. This helps when visually parsing the debug output to identify which debug instance a debug line belongs to.

#### Node.js

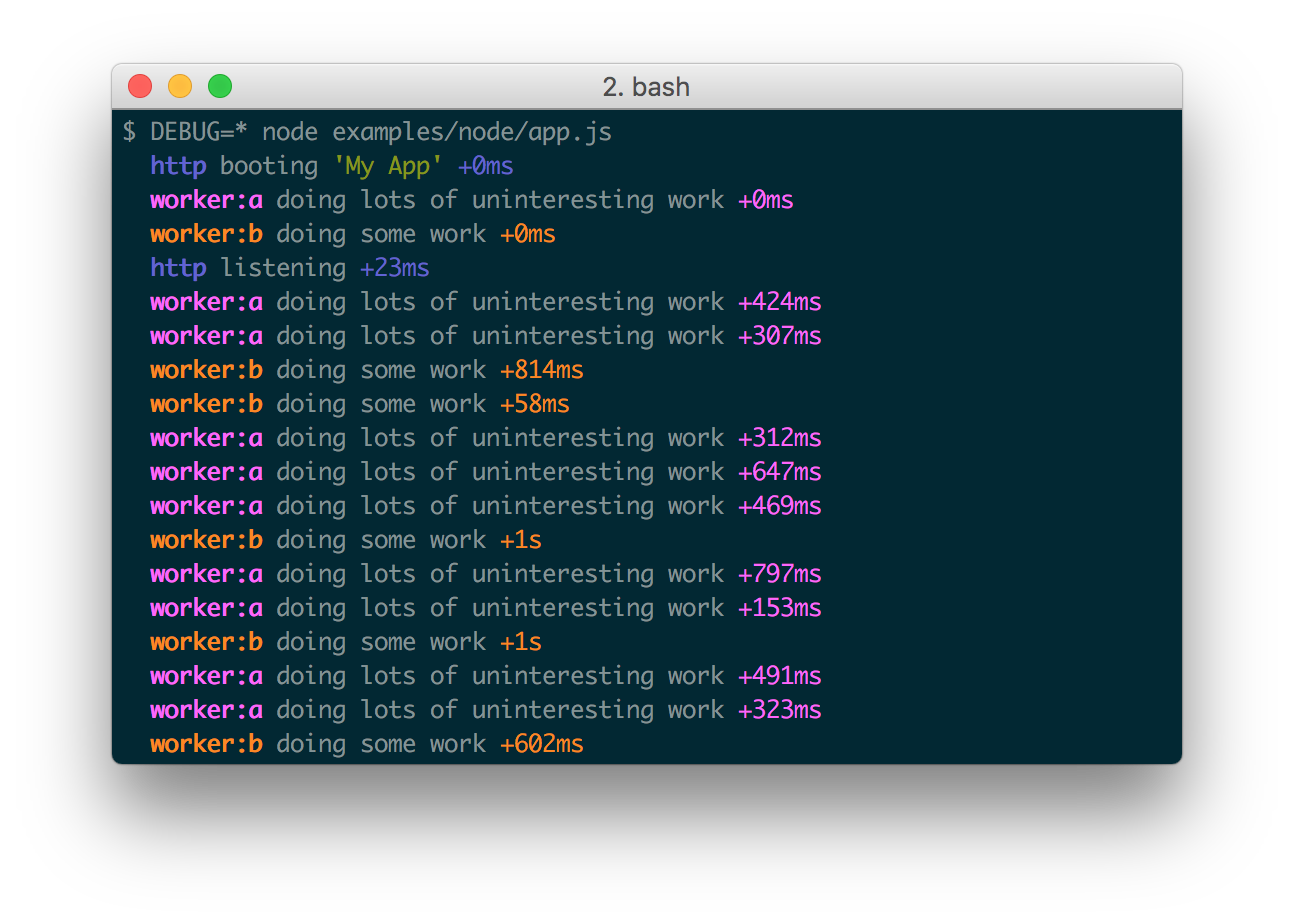
In Node.js, colors are enabled when stderr is a TTY. You also *should* install the [supports-color](https://npmjs.org/supports-color) module alongside debug, otherwise debug will only use a small handful of basic colors.

#### Web Browser

Colors are also enabled on "Web Inspectors" that understand the %c formatting option. These are WebKit web inspectors, Firefox ([since version 31](https://hacks.mozilla.org/2014/05/editable-box-model-multiple-selection-sublime-text-keys-much-more-firefox-developer-tools-episode-31/)) and the Firebug plugin for Firefox (any version).

## Millisecond diff

When actively developing an application it can be useful to see when the time spent between one debug() call and the next. Suppose for example you invoke debug() before requesting a resource, and after as well, the "+NNNms" will show you how much time was spent between calls.

 When stdout is not a TTY, Date#toISOString() is used, making it more useful for logging the debug information as shown below:

## Conventions

If you're using this in one or more of your libraries, you *should* use the name of your library so that developers may toggle debugging as desired without guessing names. If you have more than one debuggers you *should* prefix them with your library name and use ":" to separate features. For example "bodyParser" from Connect would then be "connect:bodyParser". If you append a "\*" to the end of your name, it will always be enabled regardless of the setting of the DEBUG environment variable. You can then use it for normal output as well as debug output.

## Wildcards

The \* character may be used as a wildcard. Suppose for example your library has debuggers named "connect:bodyParser", "connect:compress", "connect:session", instead of listing all three with DEBUG=connect:bodyParser,connect:compress,connect:session, you may simply do DEBUG=connect:\*, or to run everything using this module simply use DEBUG=\*.

You can also exclude specific debuggers by prefixing them with a "-" character. For example, DEBUG=\*,-connect:\* would include all debuggers except those starting with "connect:".

## Environment Variables

When running through Node.js, you can set a few environment variables that will change the behavior of the debug logging:

| **Name** | **Purpose** |
| --- | --- |
| DEBUG | Enables/disables specific debugging namespaces. |
| DEBUG\_HIDE\_DATE | Hide date from debug output (non-TTY). |
| DEBUG\_COLORS | Whether or not to use colors in the debug output. |
| DEBUG\_DEPTH | Object inspection depth. |
| DEBUG\_SHOW\_HIDDEN | Shows hidden properties on inspected objects. |

**Note:** The environment variables beginning with DEBUG\_ end up being converted into an Options object that gets used with %o/%O formatters. See the Node.js documentation for [util.inspect()](https://nodejs.org/api/util.html#util_util_inspect_object_options) for the complete list.

## Formatters

Debug uses [printf-style](https://wikipedia.org/wiki/Printf_format_string) formatting. Below are the officially supported formatters:

| **Formatter** | **Representation** |
| --- | --- |
| %O | Pretty-print an Object on multiple lines. |
| %o | Pretty-print an Object all on a single line. |
| %s | String. |
| %d | Number (both integer and float). |
| %j | JSON. Replaced with the string '[Circular]' if the argument contains circular references. |
| %% | Single percent sign ('%'). This does not consume an argument. |

### Custom formatters

You can add custom formatters by extending the debug.formatters object. For example, if you wanted to add support for rendering a Buffer as hex with %h, you could do something like:

const createDebug = require('debug')

createDebug.formatters.h = (v) => {

return v.toString('hex')

}

// …elsewhere

const debug = createDebug('foo')

debug('this is hex: %h', new Buffer('hello world'))

// foo this is hex: 68656c6c6f20776f726c6421 +0ms

## Browser Support

You can build a browser-ready script using [browserify](https://github.com/substack/node-browserify), or just use the [browserify-as-a-service](https://wzrd.in/) [build](https://wzrd.in/standalone/debug@latest), if you don't want to build it yourself.

Debug's enable state is currently persisted by localStorage. Consider the situation shown below where you have worker:a and worker:b, and wish to debug both. You can enable this using localStorage.debug:

localStorage.debug = 'worker:\*'

And then refresh the page.

a = debug('worker:a');

b = debug('worker:b');

setInterval(function(){

a('doing some work');

}, 1000);

setInterval(function(){

b('doing some work');

}, 1200);

In Chromium-based web browsers (e.g. Brave, Chrome, and Electron), the JavaScript console will—by default—only show messages logged by debug if the "Verbose" log level is *enabled*.

## Output streams

By default debug will log to stderr, however this can be configured per-namespace by overriding the log method:

Example [*stdout.js*](http://./examples/node/stdout.js):

var debug = require('debug');

var error = debug('app:error');

// by default stderr is used

error('goes to stderr!');

var log = debug('app:log');

// set this namespace to log via console.log

log.log = console.log.bind(console); // don't forget to bind to console!

log('goes to stdout');

error('still goes to stderr!');

// set all output to go via console.info

// overrides all per-namespace log settings

debug.log = console.info.bind(console);

error('now goes to stdout via console.info');

log('still goes to stdout, but via console.info now');

## Extend

You can simply extend debugger

const log = require('debug')('auth');

//creates new debug instance with extended namespace

const logSign = log.extend('sign');

const logLogin = log.extend('login');

log('hello'); // auth hello

logSign('hello'); //auth:sign hello

logLogin('hello'); //auth:login hello

## Set dynamically

You can also enable debug dynamically by calling the enable() method :

let debug = require('debug');

console.log(1, debug.enabled('test'));

debug.enable('test');

console.log(2, debug.enabled('test'));

debug.disable();

console.log(3, debug.enabled('test'));

print :

1 false

2 true

3 false

Usage :  
enable(namespaces)  
namespaces can include modes separated by a colon and wildcards.

Note that calling enable() completely overrides previously set DEBUG variable :

$ DEBUG=foo node -e 'var dbg = require("debug"); dbg.enable("bar"); console.log(dbg.enabled("foo"))'

=> false

disable()

Will disable all namespaces. The functions returns the namespaces currently enabled (and skipped). This can be useful if you want to disable debugging temporarily without knowing what was enabled to begin with.

For example:

let debug = require('debug');

debug.enable('foo:\*,-foo:bar');

let namespaces = debug.disable();

debug.enable(namespaces);

Note: There is no guarantee that the string will be identical to the initial enable string, but semantically they will be identical.

## Checking whether a debug target is enabled

After you've created a debug instance, you can determine whether or not it is enabled by checking the enabled property:

const debug = require('debug')('http');

if (debug.enabled) {

// do stuff...

}

You can also manually toggle this property to force the debug instance to be enabled or disabled.

## Usage in child processes

Due to the way debug detects if the output is a TTY or not, colors are not shown in child processes when stderr is piped. A solution is to pass the DEBUG\_COLORS=1 environment variable to the child process.  
For example:

worker = fork(WORKER\_WRAP\_PATH, [workerPath], {

stdio: [

/\* stdin: \*/ 0,

/\* stdout: \*/ 'pipe',

/\* stderr: \*/ 'pipe',

'ipc',

],

env: Object.assign({}, process.env, {

DEBUG\_COLORS: 1 // without this settings, colors won't be shown

}),

});

worker.stderr.pipe(process.stderr, { end: false });

## Authors

* TJ Holowaychuk
* Nathan Rajlich
* Andrew Rhyne
* Josh Junon

## Backers

Support us with a monthly donation and help us continue our activities. [[Become a backer](https://opencollective.com/debug#backer)]

 [](https://opencollective.com/debug/backer/1/website) [](https://opencollective.com/debug/backer/2/website) [](https://opencollective.com/debug/backer/3/website) [](https://opencollective.com/debug/backer/4/website) [](https://opencollective.com/debug/backer/5/website) [](https://opencollective.com/debug/backer/6/website) [](https://opencollective.com/debug/backer/7/website) [](https://opencollective.com/debug/backer/8/website) [](https://opencollective.com/debug/backer/9/website) [](https://opencollective.com/debug/backer/10/website) [](https://opencollective.com/debug/backer/11/website) [](https://opencollective.com/debug/backer/12/website) [](https://opencollective.com/debug/backer/13/website) [](https://opencollective.com/debug/backer/14/website) [](https://opencollective.com/debug/backer/15/website) [](https://opencollective.com/debug/backer/16/website) [](https://opencollective.com/debug/backer/17/website) [](https://opencollective.com/debug/backer/18/website) [](https://opencollective.com/debug/backer/19/website) [](https://opencollective.com/debug/backer/20/website) [](https://opencollective.com/debug/backer/21/website) [](https://opencollective.com/debug/backer/22/website) [](https://opencollective.com/debug/backer/23/website) [](https://opencollective.com/debug/backer/24/website) [](https://opencollective.com/debug/backer/25/website) [](https://opencollective.com/debug/backer/26/website) [](https://opencollective.com/debug/backer/27/website) [](https://opencollective.com/debug/backer/28/website) [](https://opencollective.com/debug/backer/29/website)

## Sponsors

Become a sponsor and get your logo on our README on Github with a link to your site. [[Become a sponsor](https://opencollective.com/debug#sponsor)]

 [](https://opencollective.com/debug/sponsor/1/website) [](https://opencollective.com/debug/sponsor/2/website) [](https://opencollective.com/debug/sponsor/3/website) [](https://opencollective.com/debug/sponsor/4/website) [](https://opencollective.com/debug/sponsor/5/website) [](https://opencollective.com/debug/sponsor/6/website) [](https://opencollective.com/debug/sponsor/7/website) [](https://opencollective.com/debug/sponsor/8/website) [](https://opencollective.com/debug/sponsor/9/website) [](https://opencollective.com/debug/sponsor/10/website) [](https://opencollective.com/debug/sponsor/11/website) [](https://opencollective.com/debug/sponsor/12/website) [](https://opencollective.com/debug/sponsor/13/website) [](https://opencollective.com/debug/sponsor/14/website) [](https://opencollective.com/debug/sponsor/15/website) [](https://opencollective.com/debug/sponsor/16/website) [](https://opencollective.com/debug/sponsor/17/website) [](https://opencollective.com/debug/sponsor/18/website) [](https://opencollective.com/debug/sponsor/19/website) [](https://opencollective.com/debug/sponsor/20/website) [](https://opencollective.com/debug/sponsor/21/website) [](https://opencollective.com/debug/sponsor/22/website) [](https://opencollective.com/debug/sponsor/23/website) [](https://opencollective.com/debug/sponsor/24/website) [](https://opencollective.com/debug/sponsor/25/website) [](https://opencollective.com/debug/sponsor/26/website) [](https://opencollective.com/debug/sponsor/27/website) [](https://opencollective.com/debug/sponsor/28/website) [](https://opencollective.com/debug/sponsor/29/website)

## License

(The MIT License)

Copyright (c) 2014-2017 TJ Holowaychuk <[tj@vision-media.ca](mailto:tj@vision-media.ca)> Copyright (c) 2018-2021 Josh Junon

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the 'Software'), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.