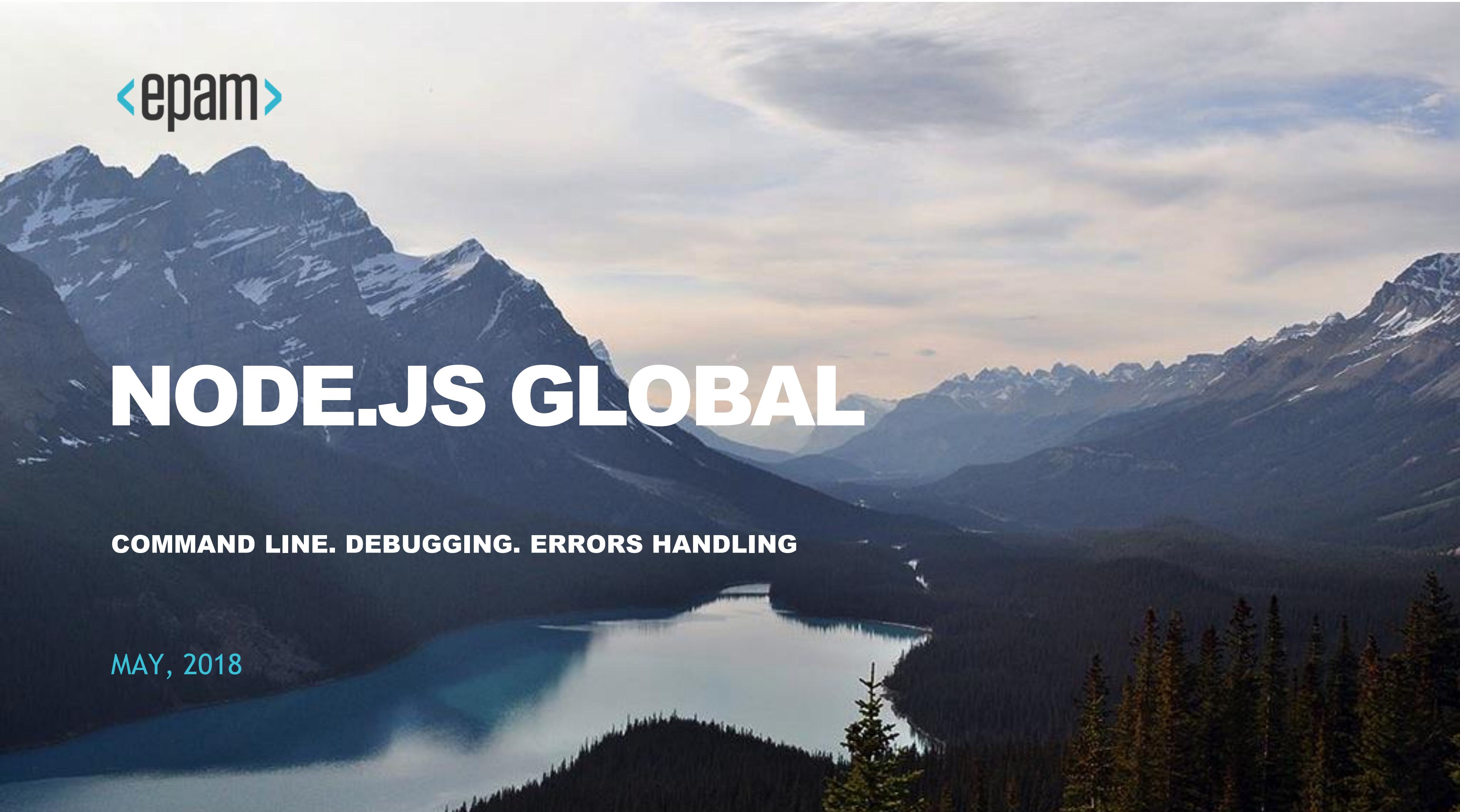


<epam>

# NODE.JS GLOBAL

COMMAND LINE. DEBUGGING. ERRORS HANDLING

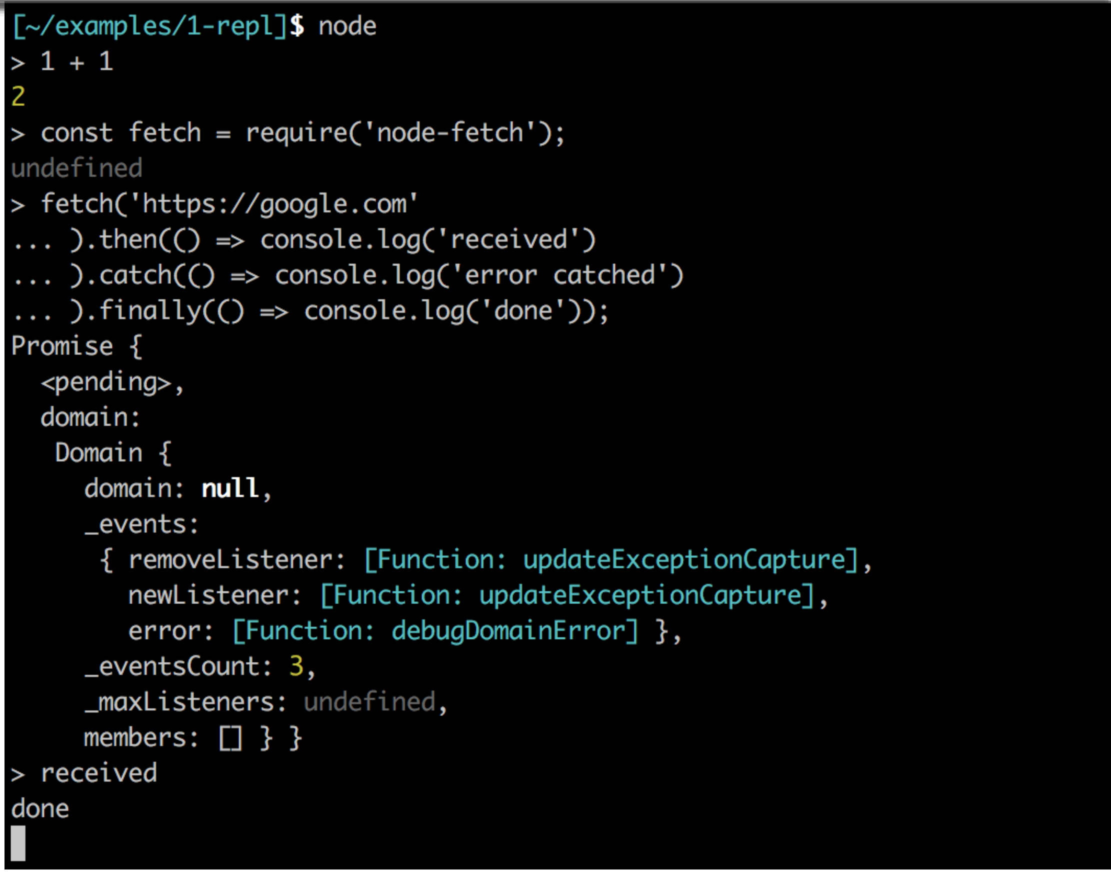
MAY, 2018

## AGENDA

- Node.JS REPL
- Command Line Tools
- Reading from the command line
- Reading environment variables
- Console
- Logging and loggers
- Debugging Node.JS
- Error handling

## NODE REPL (READ-EVALUATE-PRINT LOOP)

- REPL stands for Read Evaluate Print Loop.
- Represents an interactive environment.
- Comes together with Node.JS.

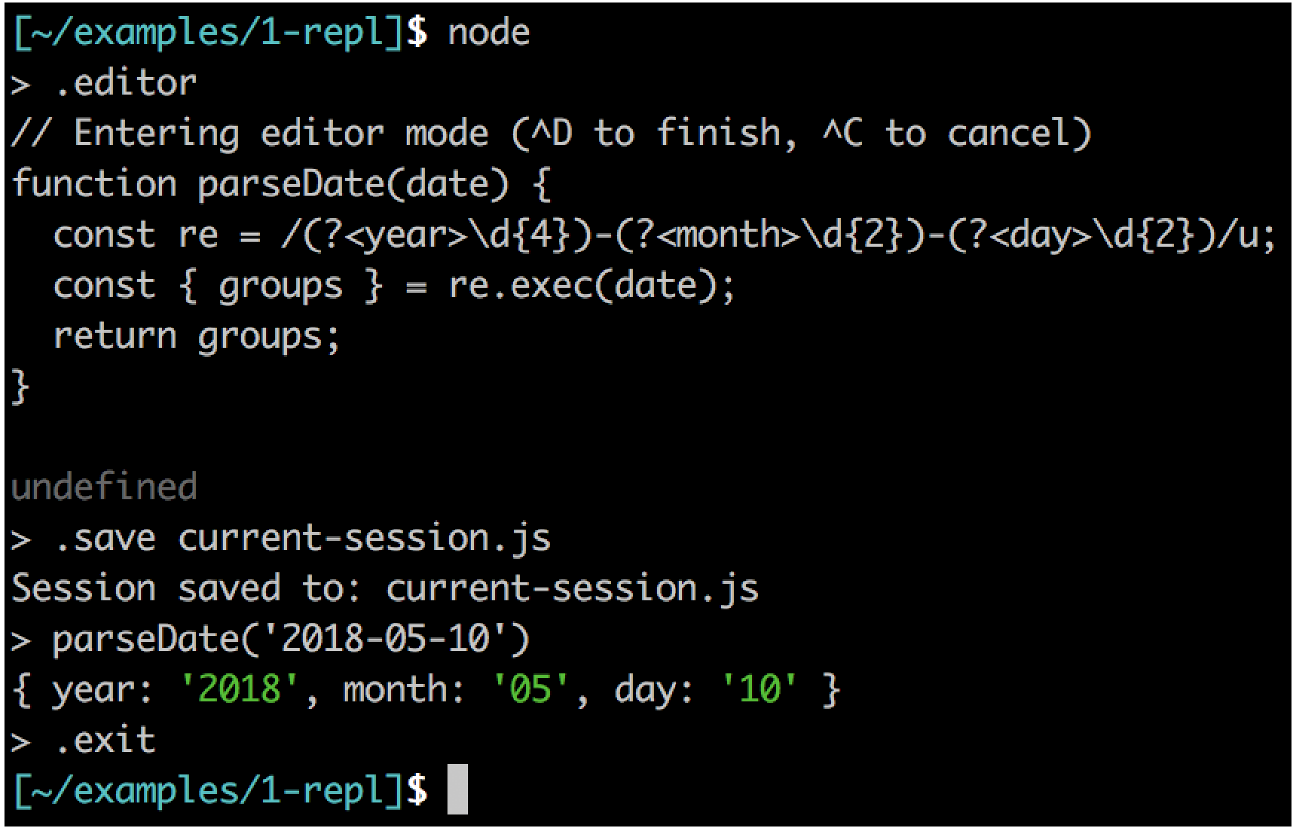


```
[~/examples/1-repl]$ node
> 1 + 1
2
> const fetch = require('node-fetch');
undefined
> fetch('https://google.com'
... ).then(() => console.log('received')
... ).catch(() => console.log('error catched')
... ).finally(() => console.log('done'));
Promise {
  <pending>,
  domain:
   Domain {
     domain: null,
     _events:
      { removeListener: [Function: updateExceptionCapture],
        newListener: [Function: updateExceptionCapture],
        error: [Function: debugDomainError] },
     _eventsCount: 3,
     _maxListeners: undefined,
     members: [] } }
> received
done
```

## NODE REPL CUSTOMIZATION

- .break
- .clear
- .exit
- .help
- .save
- .load
- .editor

- <ctrl>-C
- <ctrl>-D
- <tab>



```
[~/examples/1-repl]$ node
> .editor
// Entering editor mode (^D to finish, ^C to cancel)
function parseDate(date) {
  const re = /(?<year>\d{4})-(?<month>\d{2})-(?<day>\d{2})/u;
  const { groups } = re.exec(date);
  return groups;
}

undefined
> .save current-session.js
Session saved to: current-session.js
> parseDate('2018-05-10')
{ year: '2018', month: '05', day: '10' }
> .exit
[~/examples/1-repl]$
```

# NODE REPL CUSTOMIZATION

- Use your own eval function
- Recoverable Errors
- Output customization
- *exit* and *reset* events
- Define .-prefixed command

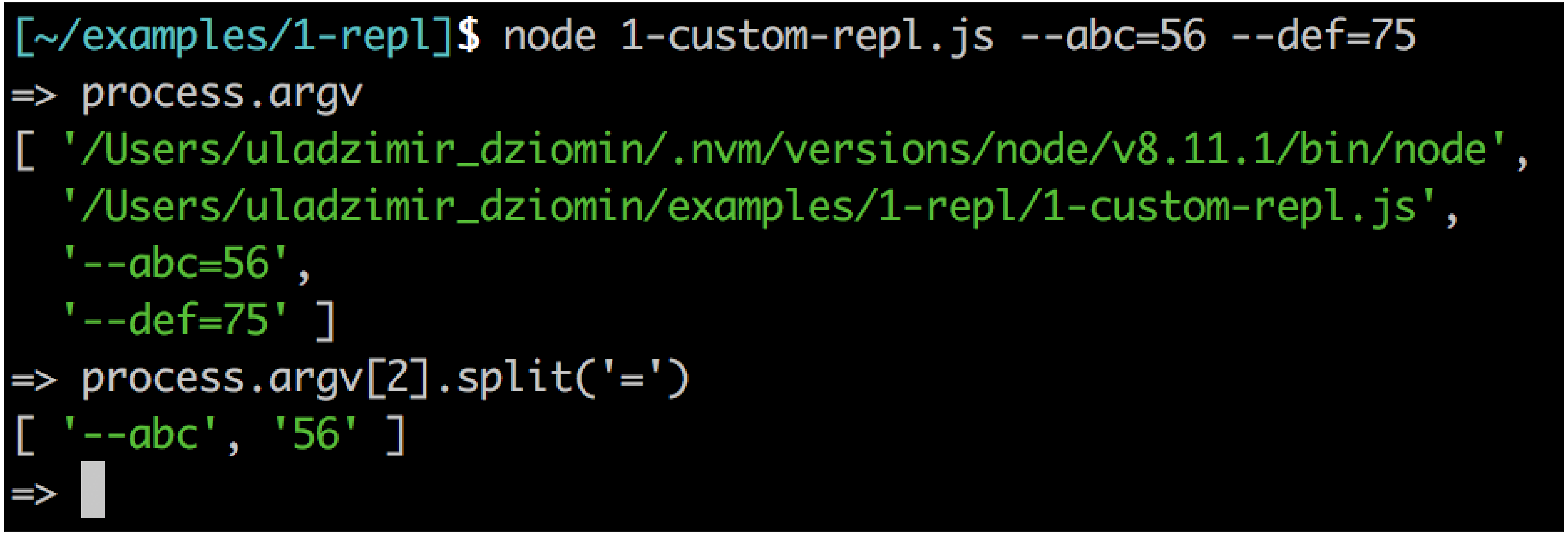
```
const repl = require('repl');
const colors = require('colors');

let replServer = repl.start({
    prompt: '༼ つ ◕_◕ ༽つ'.red,
    useColors: true,
    ignoreUndefined: true
});

replServer.context.fs = require('fs');
```

## COMMAND LINE TOOLS

**Why do we need them?**

- Same language
- Re-use application code
- Internal application utilities
- Use the same language for automation scripts

## COMMAND LINE ARGUMENTS



```
[~/examples/1-repl]$ node 1-custom-repl.js --abc=56 --def=75
=> process.argv
[ '/Users/uladzimir_dziomin/.nvm/versions/node/v8.11.1/bin/node',
  '/Users/uladzimir_dziomin/examples/1-repl/1-custom-repl.js',
  '--abc=56',
  '--def=75' ]
=> process.argv[2].split('=')
[ '--abc', '56' ]
=> 
```

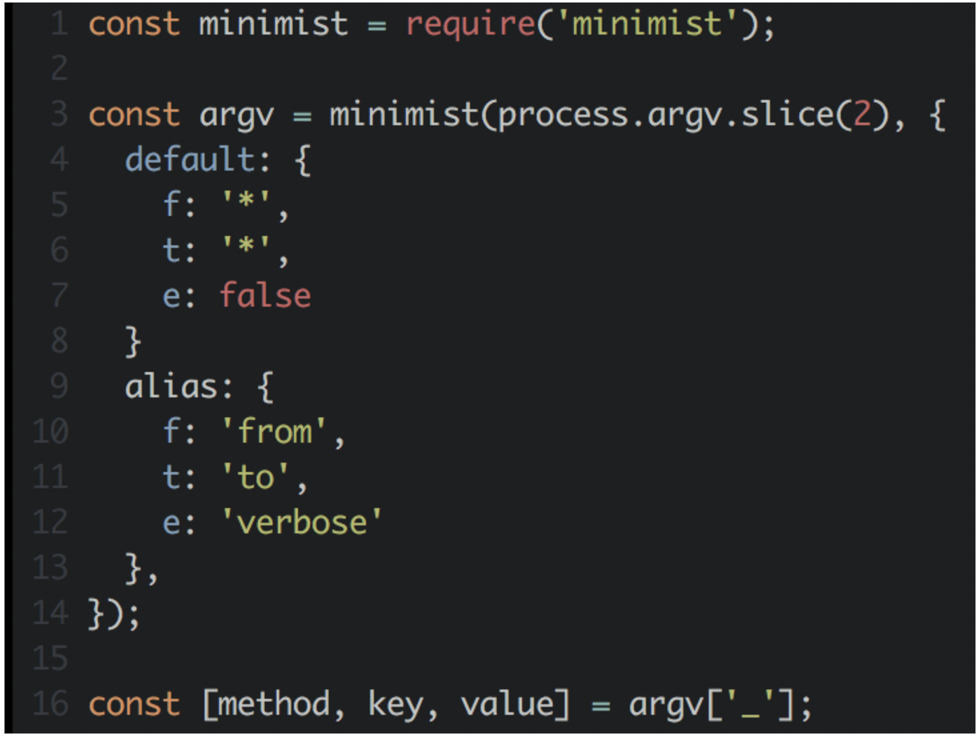
## COMMAND LINE TOOLS: COMMANDER

```
#!/usr/bin/env node

/**
 * Module dependencies.
 */

var program = require('commander');

program
  .version('0.1.0')
  .option('-p, --peppers', 'Add peppers')
  .option('-P, --pineapple', 'Add pineapple')
  .option('-b, --bbq-sauce', 'Add bbq sauce')
  .option('-c, --cheese [type]', 'Add the specified type of cheese [marbl
  .parse(process.argv);

console.log('you ordered a pizza with:');
if (program.peppers) console.log('  - peppers');
if (program.pineapple) console.log('  - pineapple');
if (program.bbqSauce) console.log('  - bbq');
console.log('  - %s cheese', program.cheese);
```

## COMMAND LINE TOOLS: COMMANDER

```
// file: ./examples/pm
var program = require('commander');

program
  .version('0.1.0')
  .command('install [name]', 'install one or more packages')
  .command('search [query]', 'search with optional query')
  .command('list', 'list packages installed', {isDefault: true})
  .parse(process.argv);
```

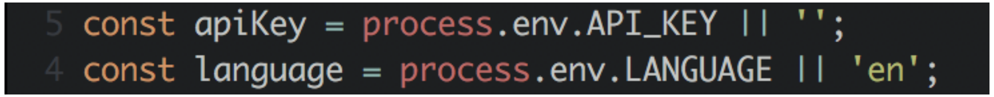
## COMMAND LINE TOOLS: MINIMIST



```
const minimist = require('minimist');

const argv = minimist(process.argv.slice(2), {
  default: {
    f: '*',
    t: '*',
    e: false
  }
  alias: {
    f: 'from',
    t: 'to',
    e: 'verbose'
  },
});

const [method, key, value] = argv['_'];
```

## ENVIRONMENT VARIABLES

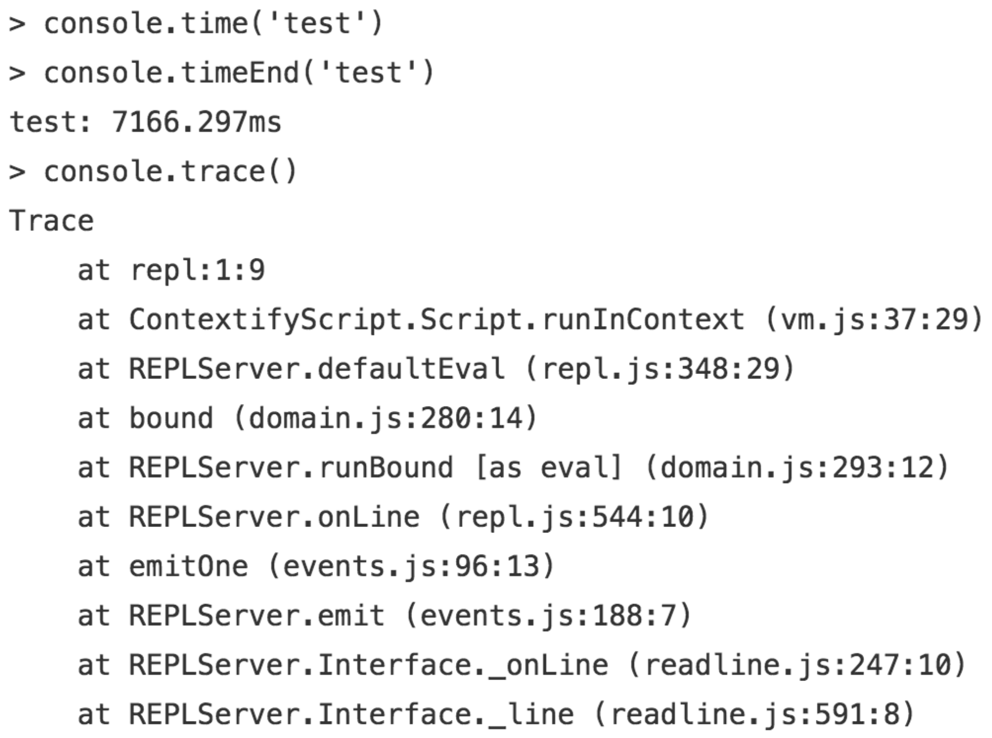
- process.env returns an object containing the user environment
- process.env modifications will not be reflected outside the Node.js process
- Assigning a property on process.env will implicitly convert the value to a string



```
const apiKey = process.env.API_KEY || '';
const language = process.env.LANGUAGE || 'en';
```

## CONSOLE

- console.dir(object, { depth: 4 })
- console.log (= console.info)
- console.error
  - console.warn
- console.assert
- console.trace
- console.time/timeEnd
- console.table



```
> console.time('test')
> console.timeEnd('test')
test: 7166.297ms
> console.trace()
Trace
    at repl:1:9
    at ContextifyScript.Script.runInContext (vm.js:37:29)
    at REPLServer.defaultEval (repl.js:348:29)
    at bound (domain.js:280:14)
    at REPLServer.runBound [as eval] (domain.js:293:12)
    at REPLServer.onLine (repl.js:544:10)
    at emitOne (events.js:96:13)
    at REPLServer.emit (events.js:188:7)
    at REPLServer.Interface._onLine (readline.js:247:10)
    at REPLServer.Interface._line (readline.js:591:8)
```

## LOGGING

- **Application metrics** – application fun
- **Errors -** resource exhaustion, uncaught exceptions, connection failures
- **Debug information -** method calls, event triggers, connections, access to resources
- **Business statistics** – logins, purchases, registrations, unsubscribes
  - Don’t break *General Data Protection Regulation* (*GDPR*)

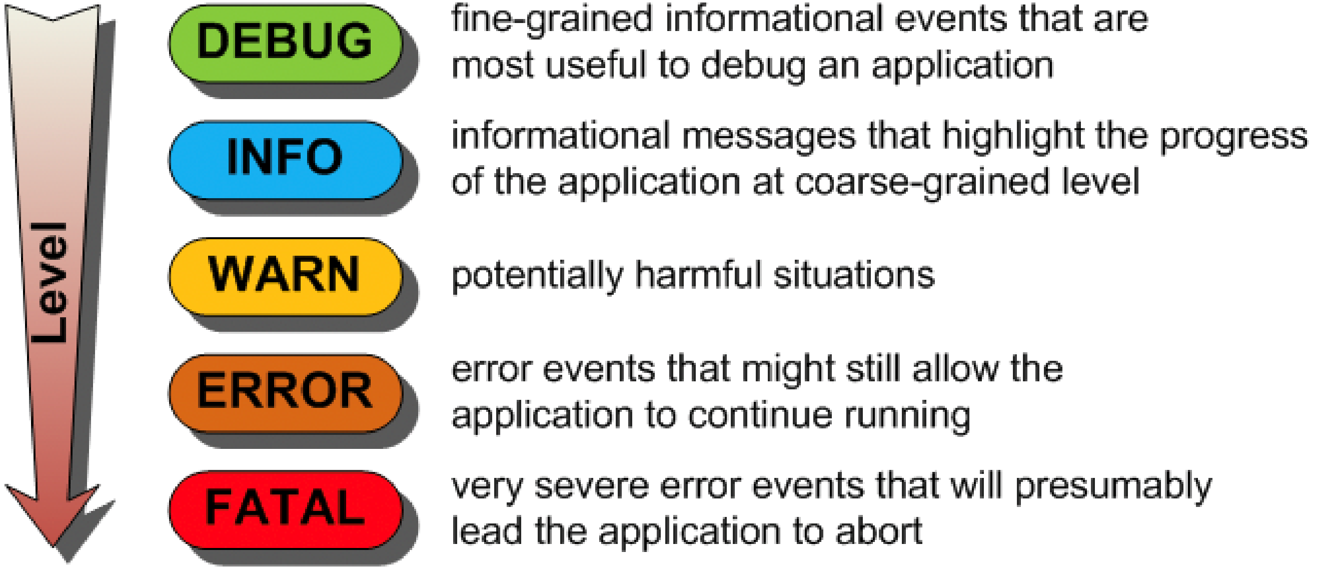
## LOGGING: RFC 5424 The Syslog Protocol

```
+---------------------+    +---------------------+
|  content            |    |  content            |
|---------------------|    |---------------------|
|  syslog application |    |  syslog application | (originator,
|                     |    |                     |  collector, relay)
|---------------------|    |---------------------|
|  syslog transport   |    |  syslog transport   | (transport sender,
|                     |    |                     | (transport receiver)
+---------------------+    +---------------------+
          ^                          ^
          |                          |
           --------------------------
```

## LOGGING: Log-levels

NPM levels:

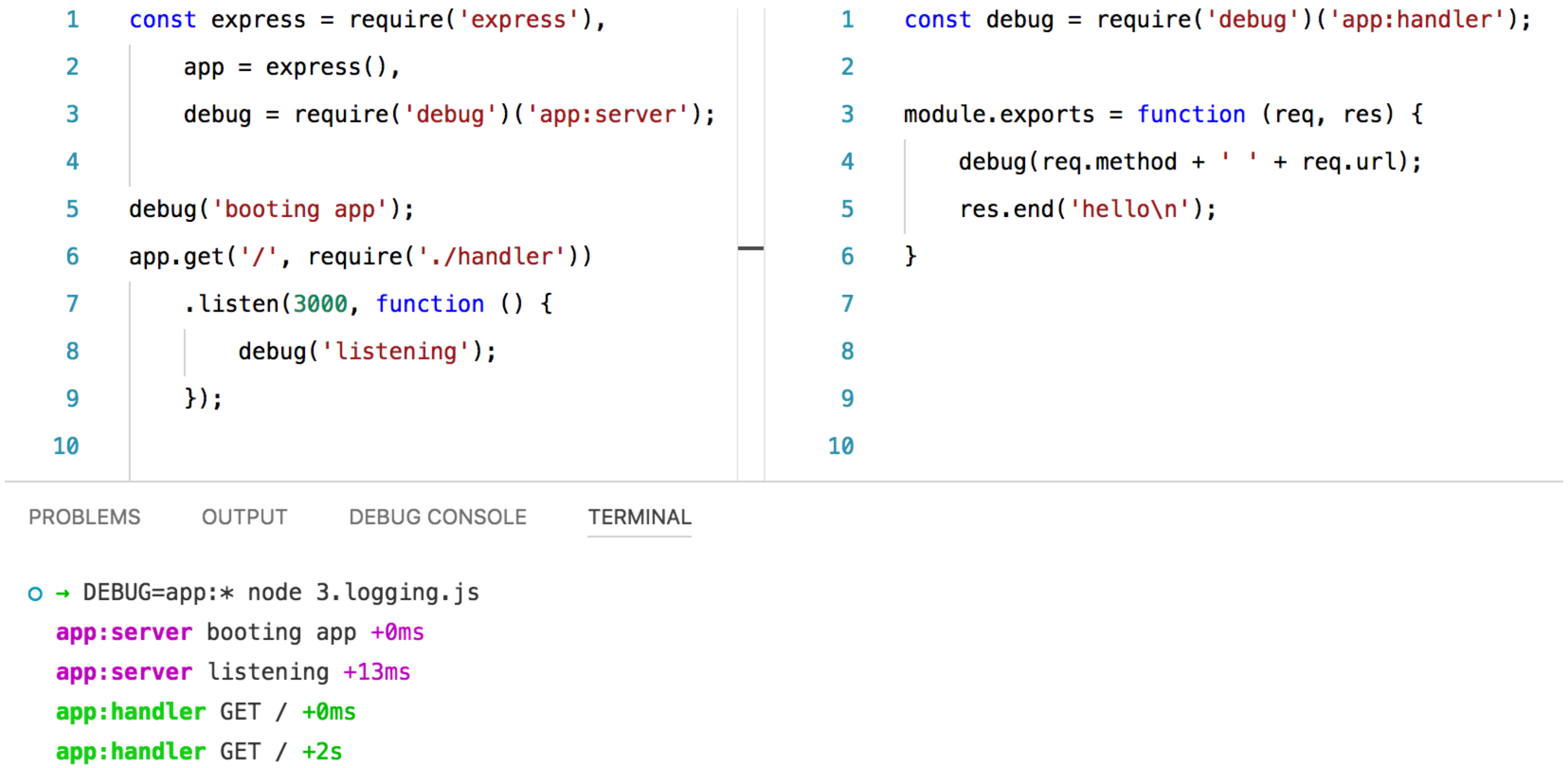
- error: 0,
- warn: 1
- info: 2
- verbose: 3
- debug: 4
- silly: 5


Level
DEBUG
fine-grained informational events that are most useful to debug an application
INFO
informational messages that highlight the progress of the application at coarse-grained level
WARN
potentially harmful situations
ERROR
error events that might still allow the application to continue running
FATAL
very severe error events that will presumably lead the application to abort

## LOGGING

- winston - Multi-transport async logging library.
- Bunyan - JSON logging library.
- Debug - The Simplest logger with minimum dependencies.

# LOGGING: DEBUG



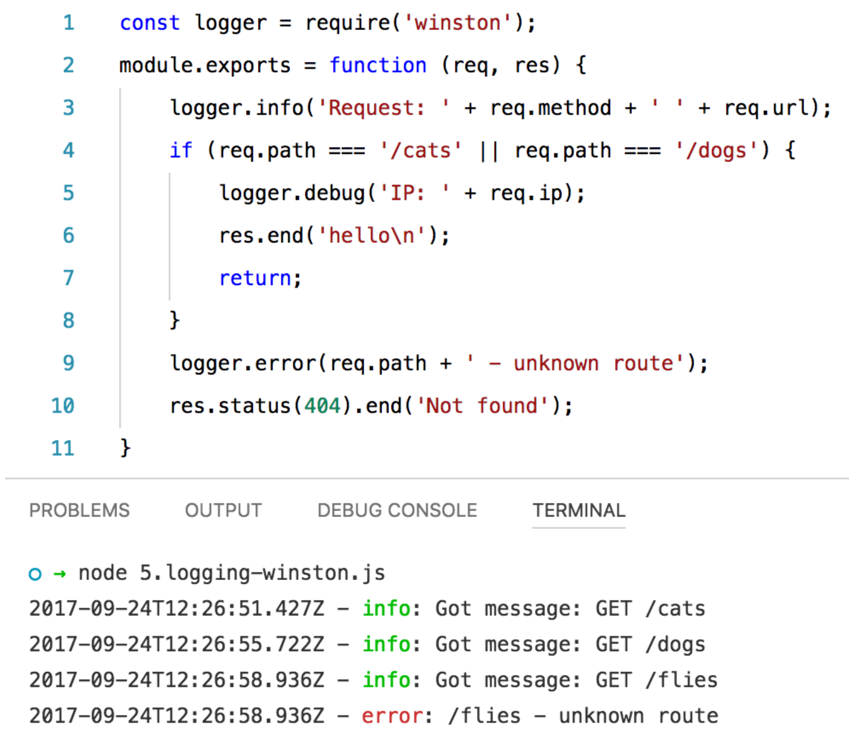
```javascript
const express = require('express'),
    app = express(),
    debug = require('debug')('app:server');

debug('booting app');
app.get('/', require('./handler'))
    .listen(3000, function () {
        debug('listening');
    });

```

```javascript
const debug = require('debug')('app:handler');

module.exports = function (req, res) {
    debug(req.method + ' ' + req.url);
    res.end('hello\n');
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
○ → DEBUG=app:* node 3.logging.js
  app:server booting app +0ms
  app:server listening +13ms
  app:handler GET / +0ms
  app:handler GET / +2s
```

## LOGGING: WINSTON



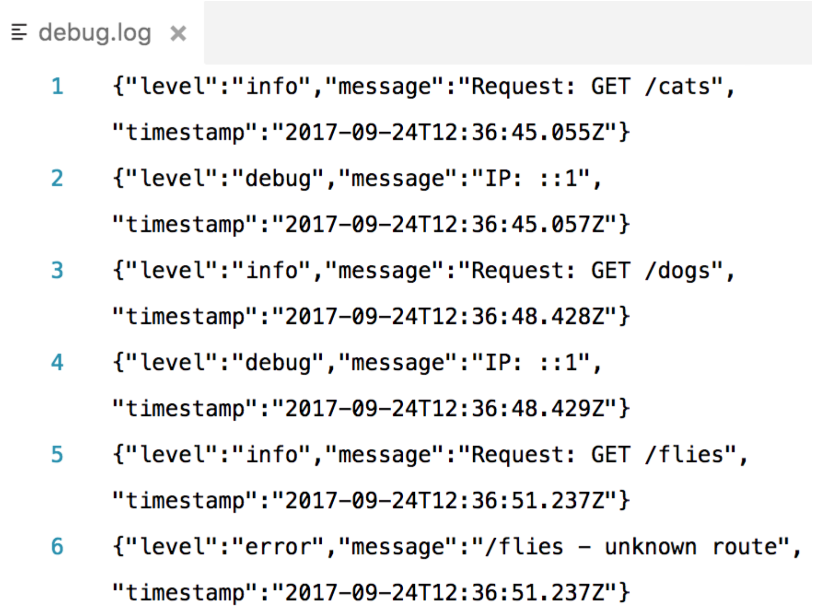
```
const logger = require('winston');
module.exports = function (req, res) {
    logger.info('Request: ' + req.method + ' ' + req.url);
    if (req.path === '/cats' || req.path === '/dogs') {
        logger.debug('IP: ' + req.ip);
        res.end('hello\n');
        return;
    }
    logger.error(req.path + ' - unknown route');
    res.status(404).end('Not found');
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
○ → node 5.logging-winston.js
2017-09-24T12:26:51.427Z - info: Got message: GET /cats
2017-09-24T12:26:55.722Z - info: Got message: GET /dogs
2017-09-24T12:26:58.936Z - info: Got message: GET /flies
2017-09-24T12:26:58.936Z - error: /flies - unknown route
```

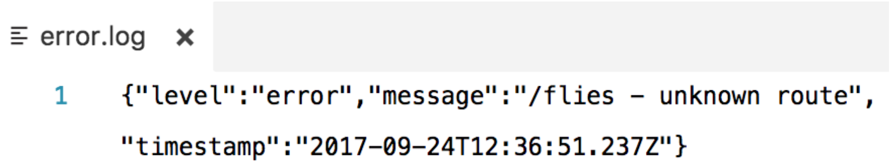
# LOGGING: WINSTON TRANSPORTS

```
let transports = [
    new winston.transports.Console({
        timestamp: true,
        colorize: true,
        level: 'info'
    }),
    new winston.transports.File({
        filename: 'debug.log',
        name: 'debug',
        level: 'debug'
    }),
    new winston.transports.File({
        filename: 'error.log',
        name: 'error',
        level: 'error'
    })];
return new winston.Logger({transports: transports});
```



debug.log

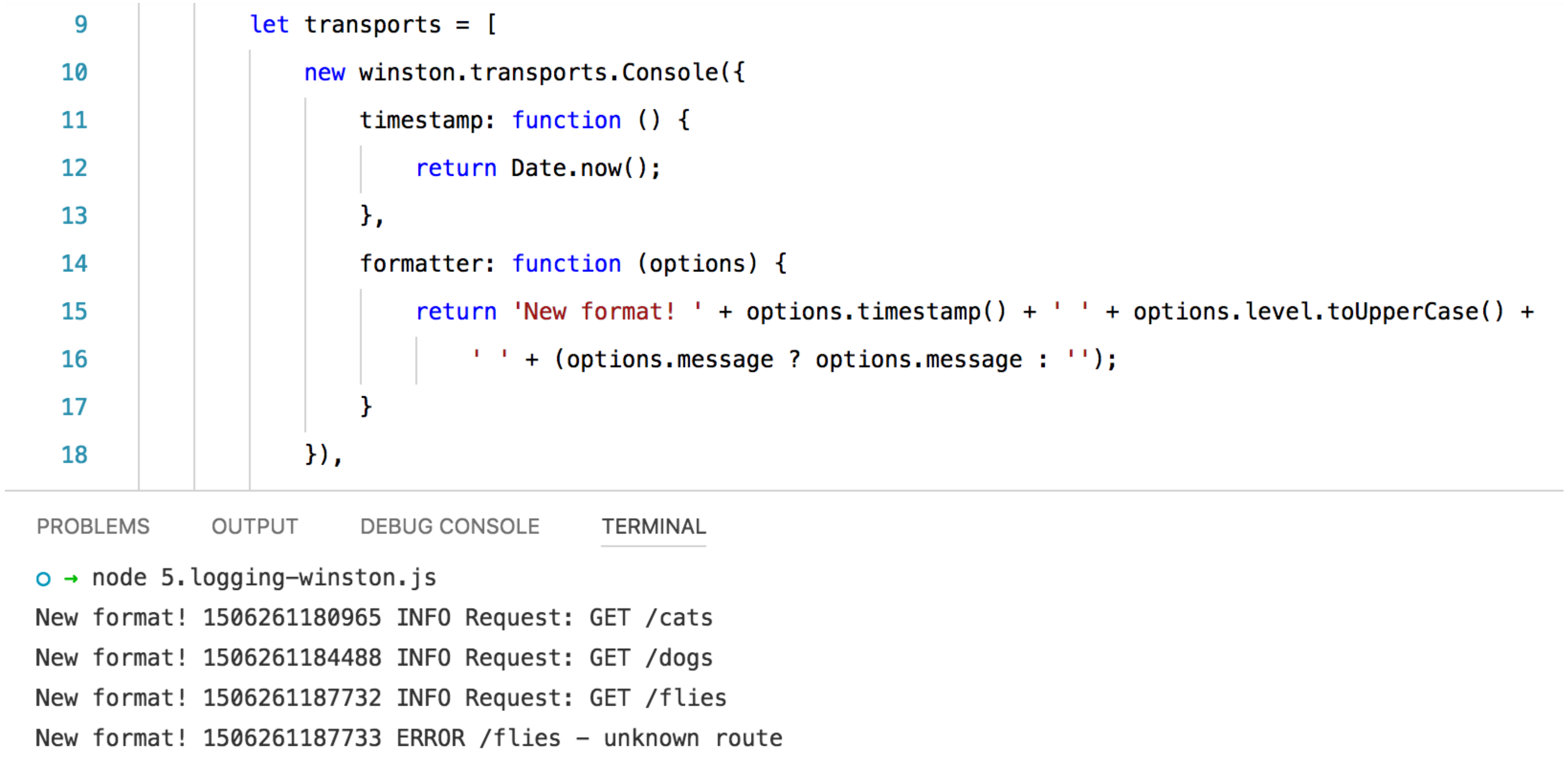
```
{"level":"info","message":"Request: GET /cats",
"timestamp":"2017-09-24T12:36:45.055Z"}
{"level":"debug","message":"IP: ::1",
"timestamp":"2017-09-24T12:36:45.057Z"}
{"level":"info","message":"Request: GET /dogs",
"timestamp":"2017-09-24T12:36:48.428Z"}
{"level":"debug","message":"IP: ::1",
"timestamp":"2017-09-24T12:36:48.429Z"}
{"level":"info","message":"Request: GET /flies",
"timestamp":"2017-09-24T12:36:51.237Z"}
{"level":"error","message":"/flies – unknown route",
"timestamp":"2017-09-24T12:36:51.237Z"}
```



error.log

```
{"level":"error","message":"/flies – unknown route",
"timestamp":"2017-09-24T12:36:51.237Z"}
```

# LOGGING: WINSTON



```
let transports = [
    new winston.transports.Console({
        timestamp: function () {
            return Date.now();
        },
        formatter: function (options) {
            return 'New format! ' + options.timestamp() + ' ' + options.level.toUpperCase() +
                ' ' + (options.message ? options.message : '');
        }
    }),
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

```
o → node 5.logging-winston.js
New format! 1506261180965 INFO Request: GET /cats
New format! 1506261184488 INFO Request: GET /dogs
New format! 1506261187732 INFO Request: GET /flies
New format! 1506261187733 ERROR /flies - unknown route
```

## LOGGING: WINSTON

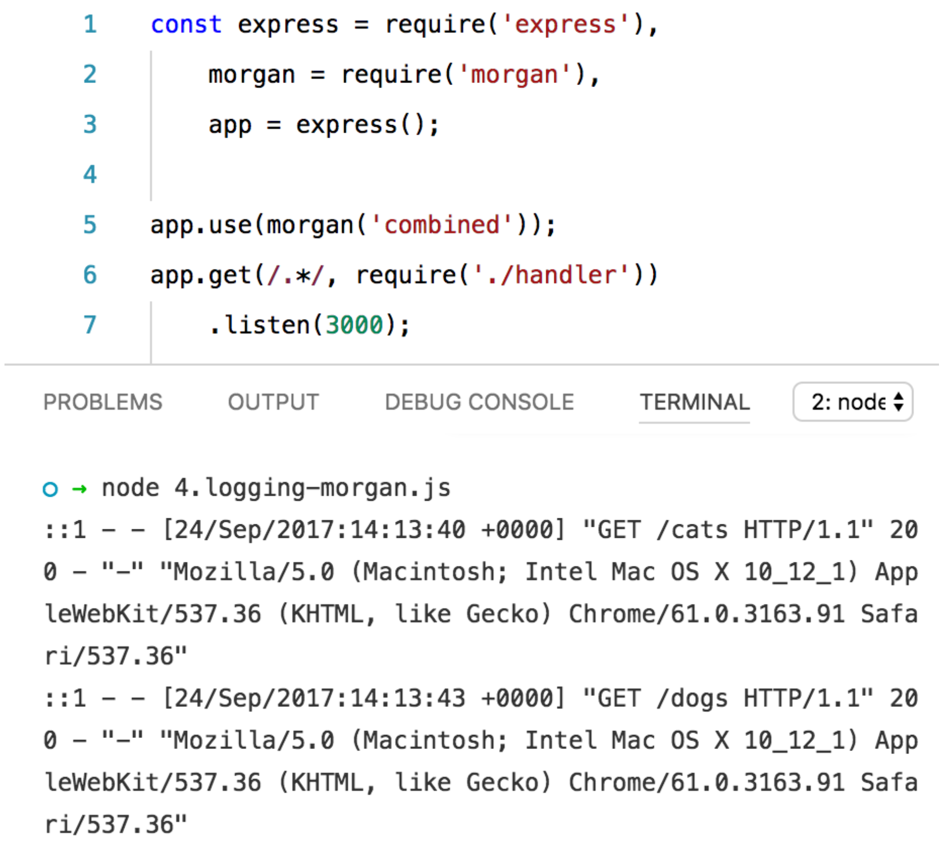
- Logging to:
  - Console,
  - Files
  - Databases (Redis, MongoDB),
  - Online services (ElasticSearch)
- Configurable logging levels
- Configurable timestamps

- Configurable output format
- Supports both string and JSON format
- Log rotation:
  - Maximum file size
  - Maximum file count
  - Zipping old files

## LOGGING: MORGAN

- Used as middleware with HTTP servers.
- Creates access logs.
- Configurable format using predefined tokens.
- Can log to any writable stream.
- Log rotation using rotating-file-stream.

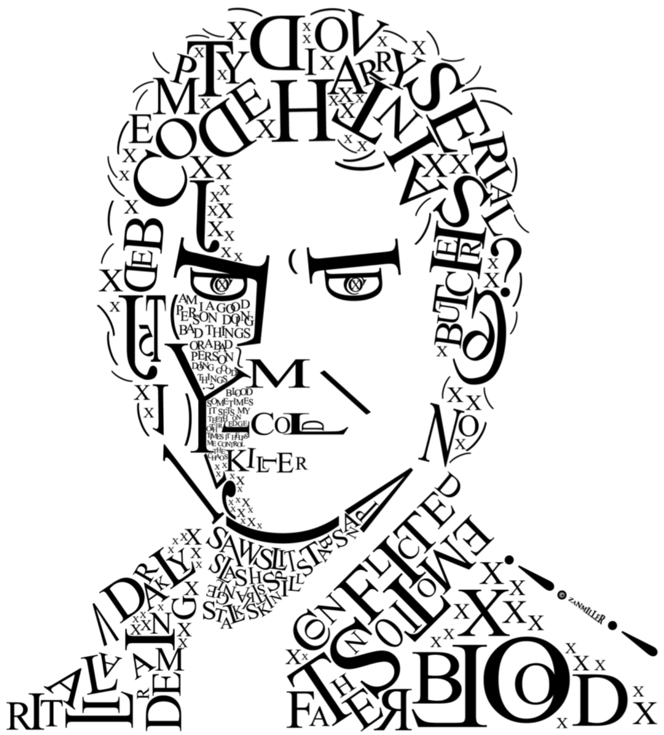
# LOGGING: MORGAN



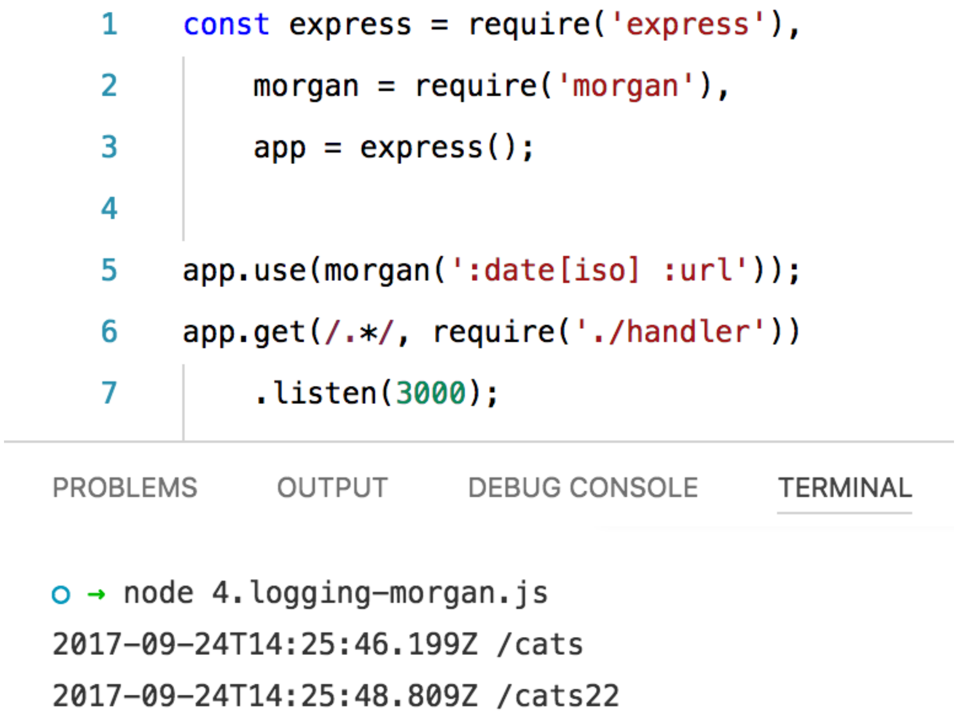
```javascript
const express = require('express'),
    morgan = require('morgan'),
    app = express();

app.use(morgan('combined'));
app.get(/.*/, require('./handler'))
    .listen(3000);
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    2: node

```
○ → node 4.logging-morgan.js
::1 - - [24/Sep/2017:14:13:40 +0000] "GET /cats HTTP/1.1" 20
0 - "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_1) App
leWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.91 Safa
ri/537.36"
::1 - - [24/Sep/2017:14:13:43 +0000] "GET /dogs HTTP/1.1" 20
0 - "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_1) App
leWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.91 Safa
ri/537.36"
```

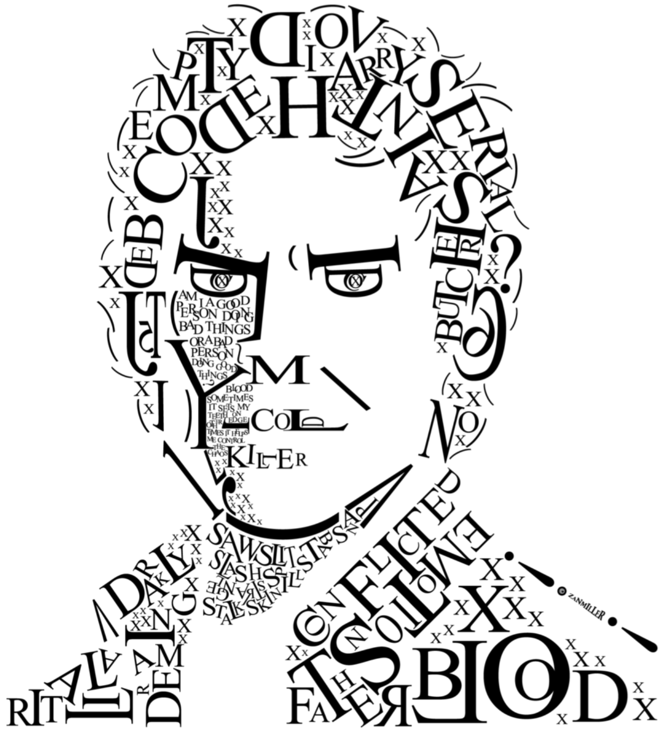


## LOGGING: MORGAN



```
const express = require('express'),
    morgan = require('morgan'),
    app = express();

app.use(morgan(':date[iso] :url'));
app.get(/.*/, require('./handler'))
    .listen(3000);
```

```
○ → node 4.logging-morgan.js
2017-09-24T14:25:46.199Z /cats
2017-09-24T14:25:48.809Z /cats22
```



## LOGGING: MORGAN

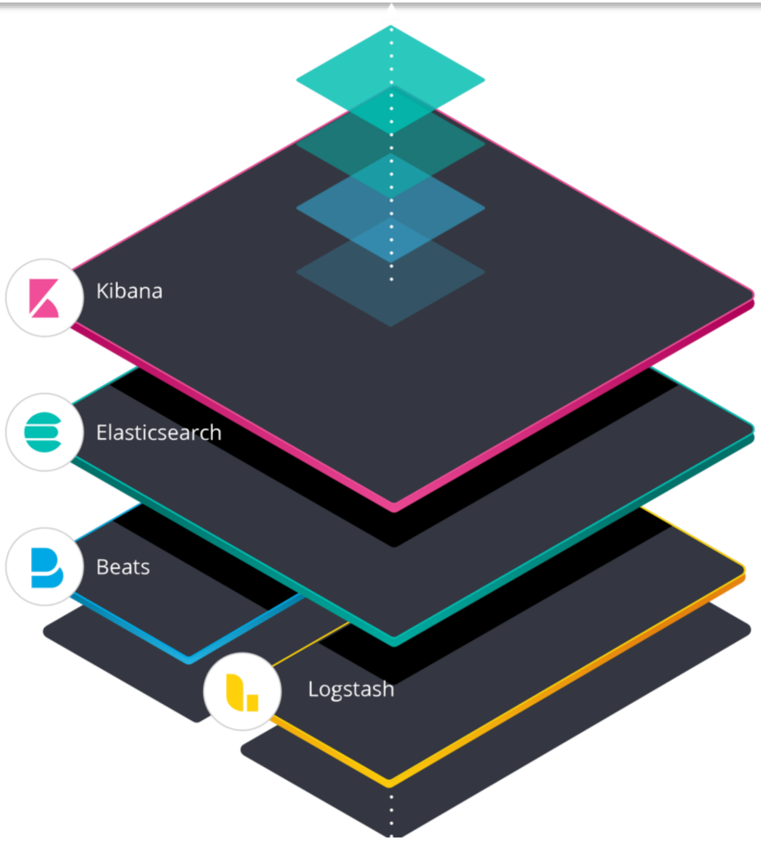
- :date[format]
    - CLF for the common log format
    - ISO (ISO 8601)
    - web (default, RFC 1123)
- :http-version
- :method
- :referrer
- :remote-addr
- :remote-user
- :req[header]
- :res[header]
- :response-time[digits] • :status
- :url
- :user-agent

## LOGGING: WINSTON

- Logging to:
  - Console,
  - Files
  - Databases (Redis, MongoDB),
  - Online services (ElasticSearch)
- Configurable logging levels
- Configurable timestamps

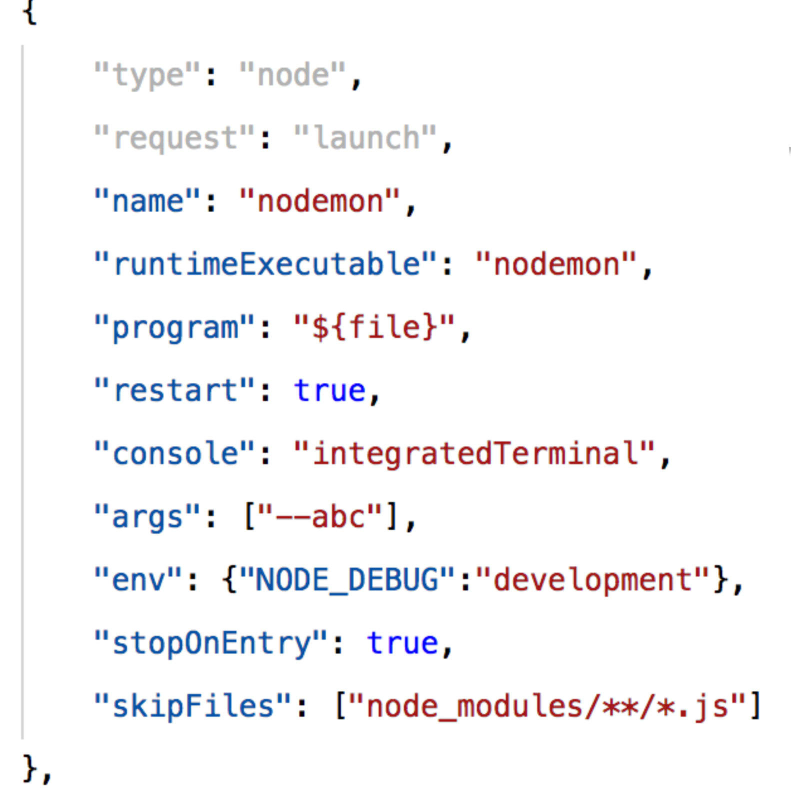
- Configurable output format
- Supports both string and JSON format
- Log rotation:
  - Maximum file size
  - Maximum file count
  - Zipping old files

## LOGGING: Log Collection

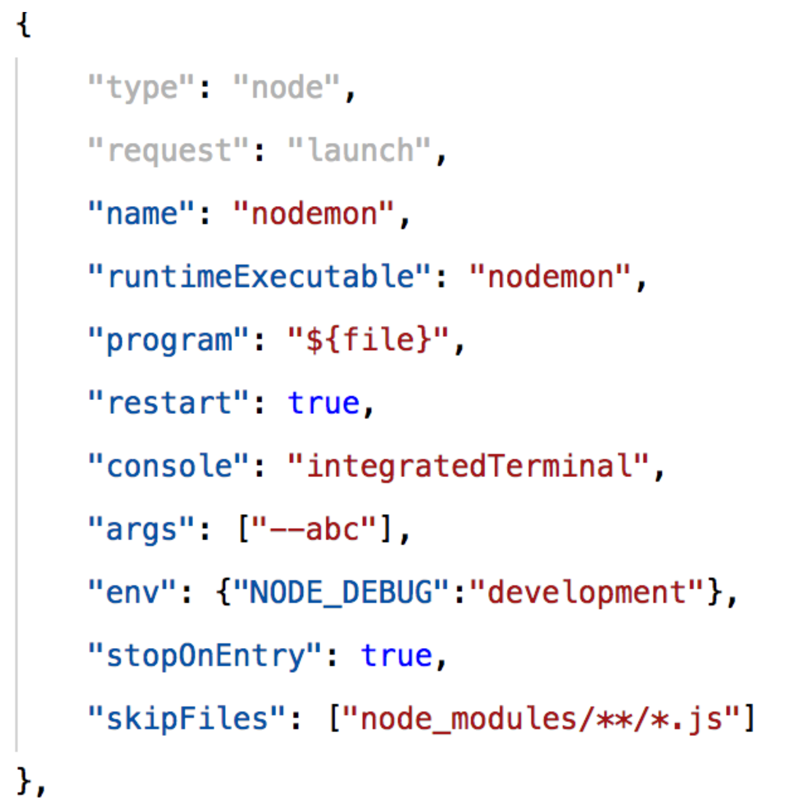


# DEBUGGING

- VSCode
- Inspect and Node debugger
- lldb + llnode

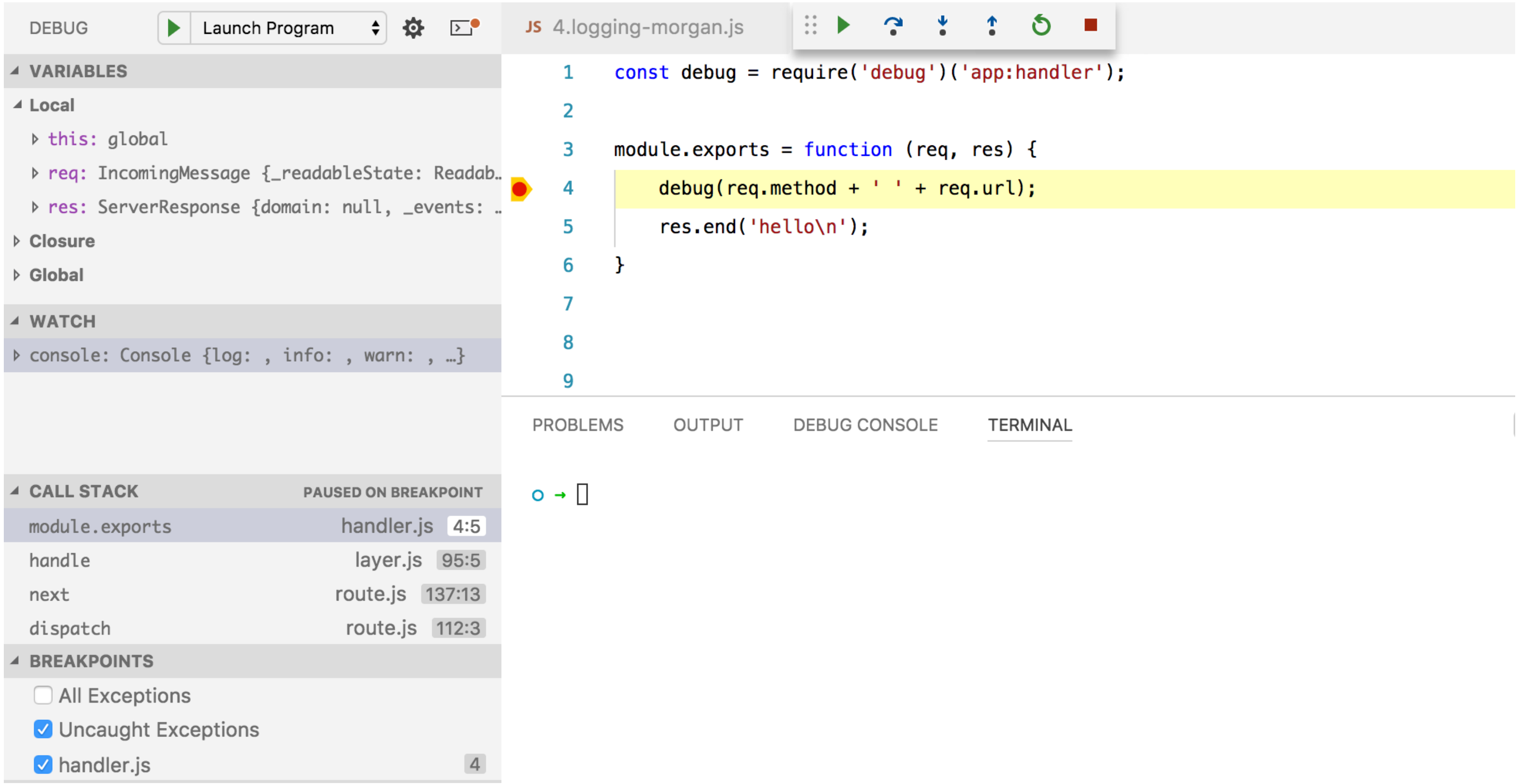


```
{
    "type": "node",
    "request": "launch",
    "name": "nodemon",
    "runtimeExecutable": "nodemon",
    "program": "${file}",
    "restart": true,
    "console": "integratedTerminal",
    "args": ["--abc"],
    "env": {"NODE_DEBUG":"development"},
    "stopOnEntry": true,
    "skipFiles": ["node_modules/**/*.js"]
},
```



```
{
    "type": "node",
    "request": "launch",
    "name": "nodemon",
    "runtimeExecutable": "nodemon",
    "program": "${file}",
    "restart": true,
    "console": "integratedTerminal",
    "args": ["--abc"],
    "env": {"NODE_DEBUG":"development"},
    "stopOnEntry": true,
    "skipFiles": ["node_modules/**/*.js"]
},
```

# DEBUGGING: VSCODE



```
this: global
req: IncomingMessage {_readableState: Readab…
res: ServerResponse {domain: null, _events: …
```

Closure

Global

```
console: Console {log: , info: , warn: , …}
```

| | |
|---|---|
| module.exports | handler.js 4:5 |
| handle | layer.js 95:5 |
| next | route.js 137:13 |
| dispatch | route.js 112:3 |

- [ ] All Exceptions
- [x] Uncaught Exceptions
- [x] handler.js 4

```javascript
const debug = require('debug')('app:handler');

module.exports = function (req, res) {
    debug(req.method + ' ' + req.url);
    res.end('hello\n');
}
```

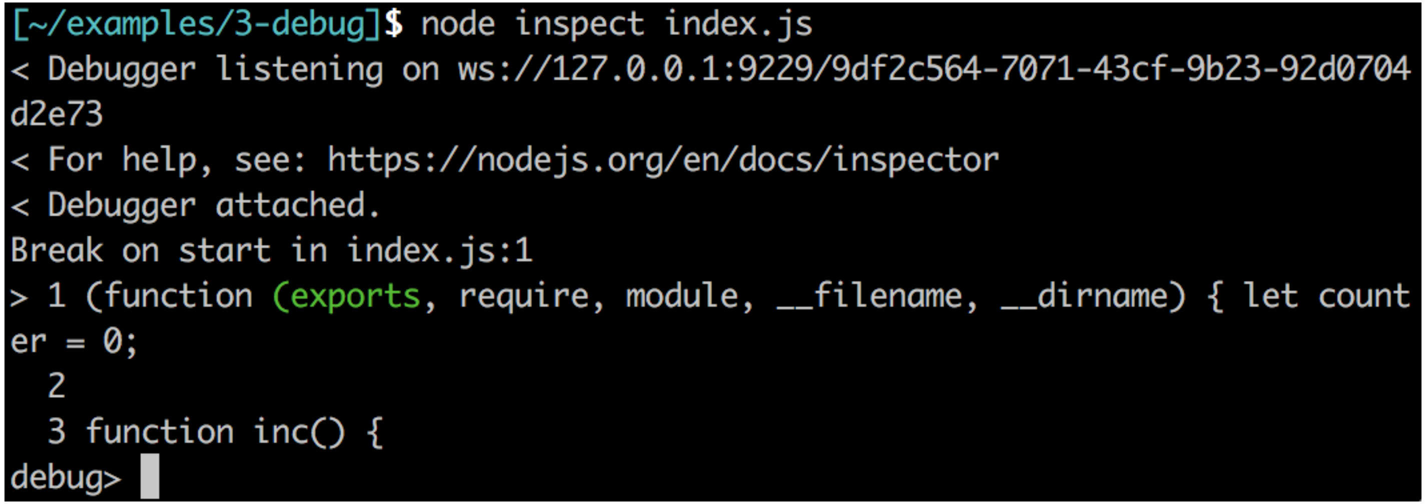
```
o → 
```

## DEBUGGING: INSPECT

V8 Inspector integration allows attaching Chrome DevTools to Node.js instances for debugging and profiling. It uses the Chrome DevTools Protocol.

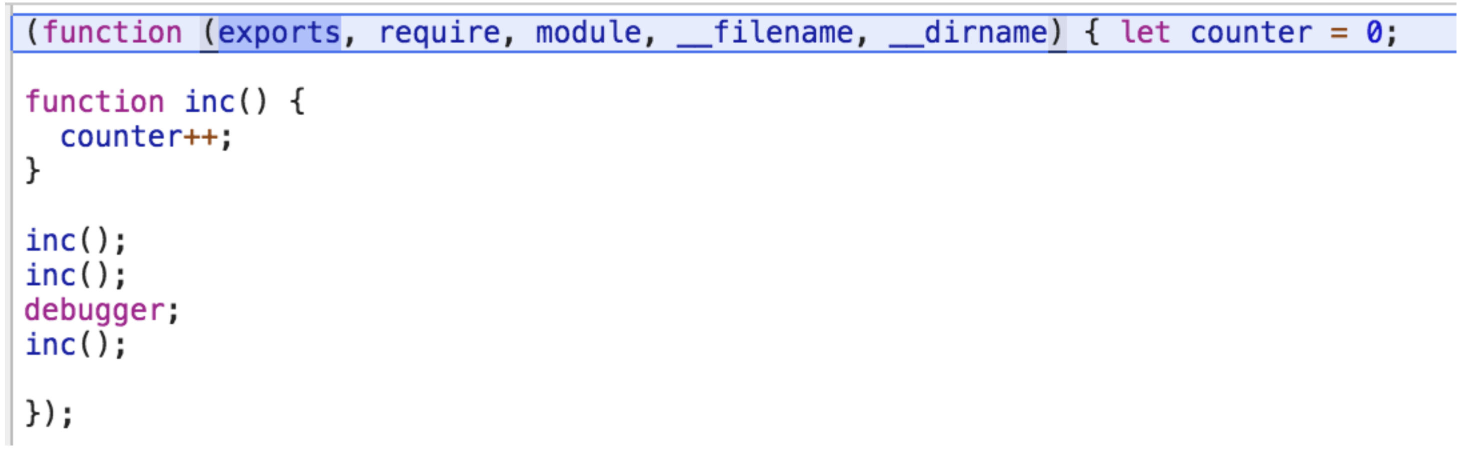
- Install Chrome extension: Node.js V8 --inspector Manager (NiM)
- Set *debugger;* instruction to stop execution on the specific line (optional)
- Run your application with inspect
- Debug in common Chrome Dev Tools environment

## DEBUGGING: INSPECT



```
[~/examples/3-debug]$ node inspect index.js
< Debugger listening on ws://127.0.0.1:9229/9df2c564-7071-43cf-9b23-92d0704
d2e73
< For help, see: https://nodejs.org/en/docs/inspector
< Debugger attached.
Break on start in index.js:1
> 1 (function (exports, require, module, __filename, __dirname) { let count
er = 0;
  2
  3 function inc() {
debug>
```

## DEBUGGING: INSPECT in CHROME DEV TOOLS



```
(function (exports, require, module, __filename, __dirname) { let counter = 0;

function inc() {
  counter++;
}

inc();
inc();
debugger;
inc();

});
```

## DEBUGGING: Debugging on remote server

[NOT FOR PRODUCTION USE]

- Bind port
    - ssh –N –i <ssh-key> -L 9229:127.0.0.1:9229
- kill –SIGUSR1 <node-app-id>
- Connect with Chrome Dev Tools

## DEBUGGING: lldb + llnode

- Node.js will create core dump of process memory snapshot
  - node –abort-on-uncaught-exception app.js
- Debug with lldb:
  - llnode –c /cores/core.12452

## ERRORS HANDLING: ERRORS

- Standard JavaScript errors
- System errors triggered by underlying operating system
- User-specified errors
- *AssertionError*

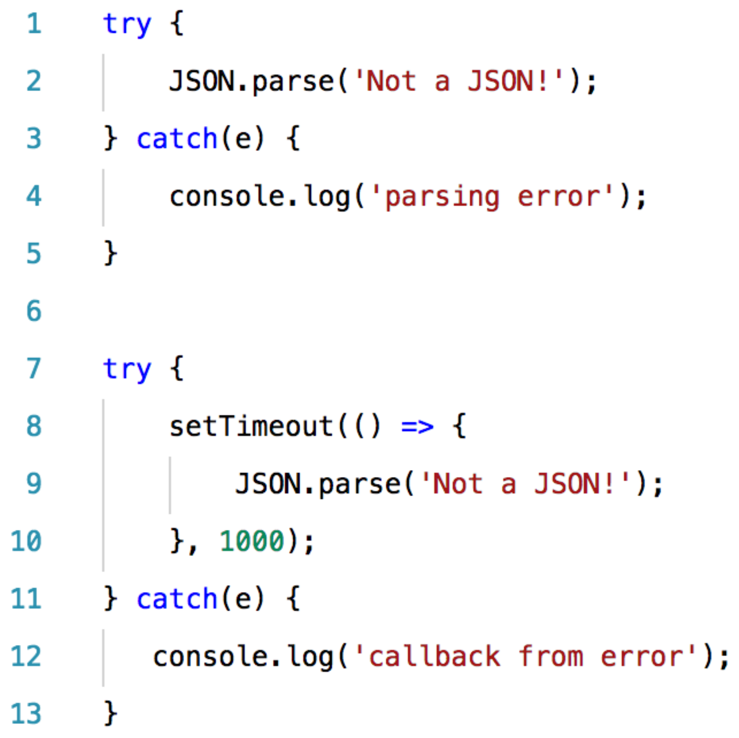
## ERRORS HANDLING: HANDLING

- try / catch construct
- Error-first callbacks
- *error* event

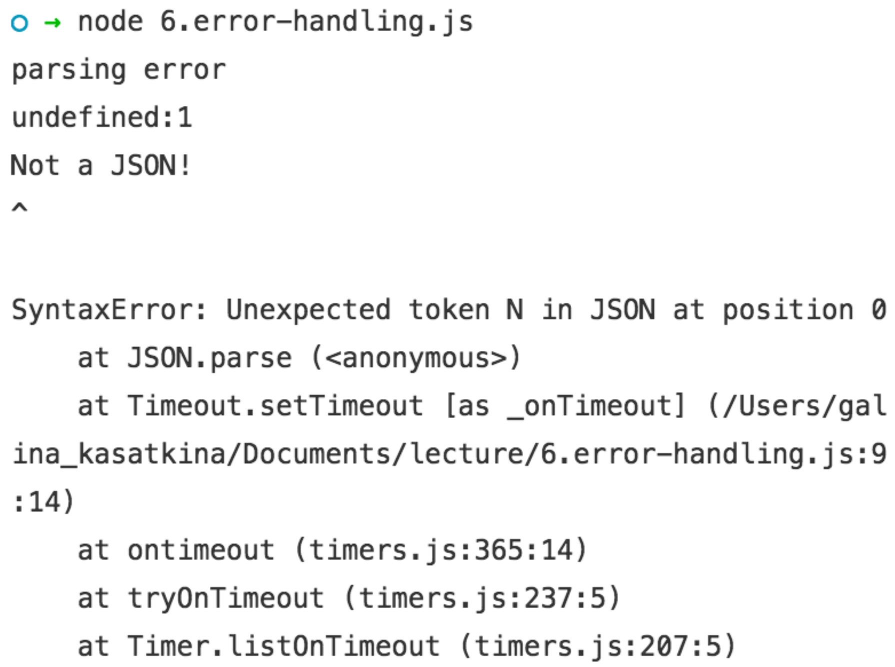
## ERROR HANDLING: ERROR CODES

- EACCES - permission denied
- EADDRINUSE - address already in use
- ECONNREFUSED - connection refused
- ECONNRESET - connection reset by peer • EEXIST - file exists
- EISDIR - is a directory
- ENOTDIR – not a directory
- ENOENT – no such file or directory
- And others

# ERROR HANDLING: TRY-CATCH



```
try {
    JSON.parse('Not a JSON!');
} catch(e) {
    console.log('parsing error');
}

try {
    setTimeout(() => {
        JSON.parse('Not a JSON!');
    }, 1000);
} catch(e) {
   console.log('callback from error');
}
```



```
o → node 6.error-handling.js
parsing error
undefined:1
Not a JSON!
^

SyntaxError: Unexpected token N in JSON at position 0
    at JSON.parse (<anonymous>)
    at Timeout.setTimeout [as _onTimeout] (/Users/gal
ina_kasatkina/Documents/lecture/6.error-handling.js:9
:14)
    at ontimeout (timers.js:365:14)
    at tryOnTimeout (timers.js:237:5)
    at Timer.listOnTimeout (timers.js:207:5)
```

## ERROR HANDLING: ERROR-FIRST CALLBACK

```
const fs = require('fs');
fs.readFile('nonexistent', (err, data) => {
    if (err) {
        console.log(err);
        return;
    }
    //do sth
});
```

## ERROR HANDLING: PROMISES

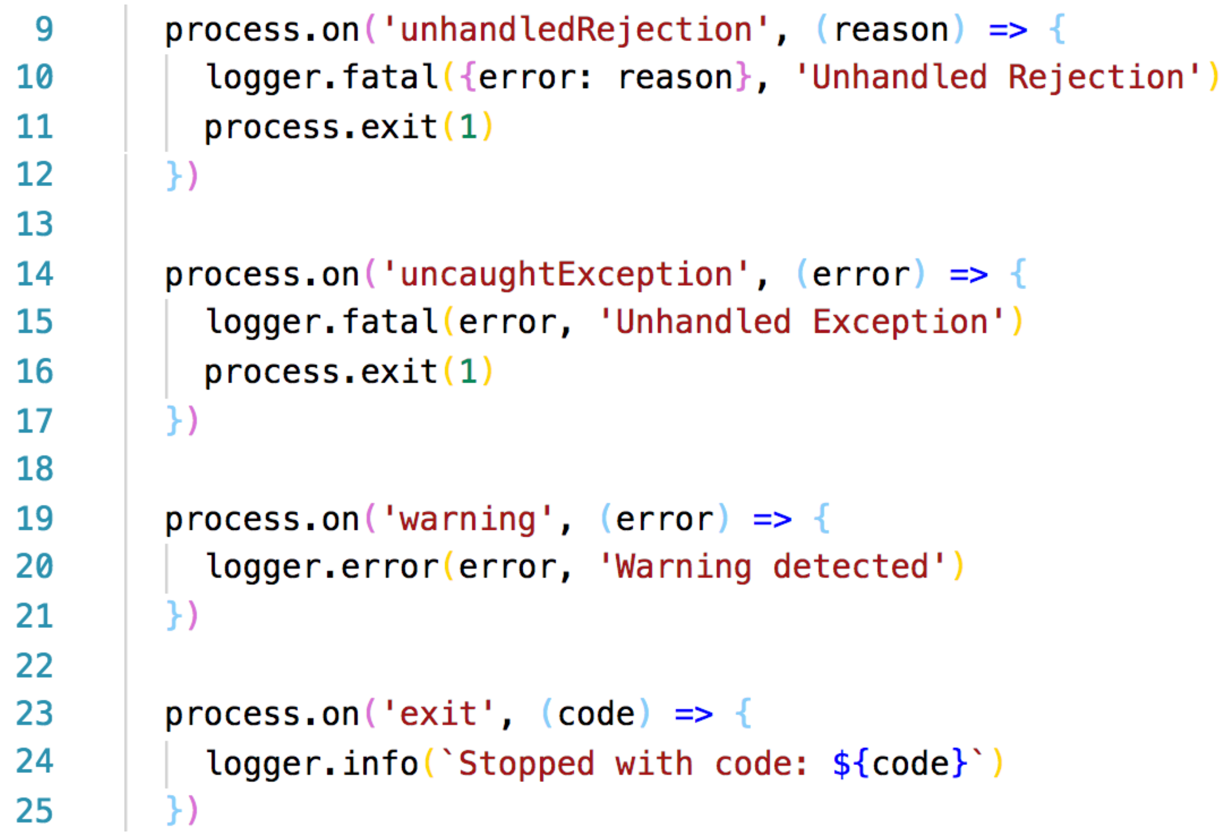
```javascript
const promisify = require("util").promisify;
const fs = require('fs');
const readFile = promisify(fs.readFile);

readFile('nonexistent')
    .then((data) => {
        JSON.parse(data);
    }).catch((err) => {
        console.log(err);
    });
```

## ERROR HANDLING: ERROR EVENTS

```javascript
const http = require('http');
const server = http.createServer((req, res) => {
  res.end('Hello!')
});

server.on('error', (err) => {
    console.error('ERROR!!!');
    console.error(err);
});

```

```
ERROR!!!
{ Error: listen EACCES 0.0.0.0:80
    at Object._errnoException (util.js:1026:11)
    at _exceptionWithHostPort (util.js:1049:20)
    at Server.setupListenHandle [as _listen2] (net.js:1
326:19)
    at listenInCluster (net.js:1391:12)
    at Server.listen (net.js:1474:7)
    at Object.<anonymous> (/Users/galina_kasatkina/Docu
ments/lecture/7.error-handling-events.1.js:11:8)
    at Module._compile (module.js:624:30)
    at Object.Module._extensions..js (module.js:635:10)
    at Module.load (module.js:545:32)
    at tryModuleLoad (module.js:508:12)
  code: 'EACCES',
  errno: 'EACCES',
  syscall: 'listen',
  address: '0.0.0.0',
  port: 80 }
```

# ERROR HANDLING: ERROR EVENTS



```javascript
process.on('unhandledRejection', (reason) => {
  logger.fatal({error: reason}, 'Unhandled Rejection')
  process.exit(1)
})

process.on('uncaughtException', (error) => {
  logger.fatal(error, 'Unhandled Exception')
  process.exit(1)
})

process.on('warning', (error) => {
  logger.error(error, 'Warning detected')
})

process.on('exit', (code) => {
  logger.info(`Stopped with code: ${code}`)
})
```

## ERROR HANDLING: UNCAUGHT EXCEPTIONS

- What SHOULD be NOT done in the handler:
  - Attempt to restore the program’s normal operation
- What you SHOULD do:
  - Log errors,
  - Free all resources,
  - Exit the process with an appropriate error code.

# ERROR HANDLING: GRACEFULL SHUTDOWN

```
  ['SIGTERM', 'SIGINT', 'SIGHUP'].forEach((sigEvent) => {
    process.on(sigEvent, () => this.stop())
  })
```

```
async function stop () {
  logger.info('Stopping...');

  const timeoutId = setTimeout(() => {
    logger.error('Stopped forcefully, cleaning Event Loop');
    process.exit(1);
  }, settings.shutdownTimeout);

  try {
    await shutdownApp();
    timeoutId.unref();
  } catch (error) {
    logger.error(error, 'Error during shutdown');
    process.exit(1);
  }
}
```

## ERROR HANDLING: CUSTOM ERROR TYPES

```
const dictionary = { hello: "Hello", world: "World" };
const logger = require('winston');

class DictError extends Error {
    constructor(word) {
        super(word);
        logger.warn('Missing translation for: ' + word);
    }
}
```

## USEFUL LINKS

- Error handling in Node: https://www.joyent.com/node-js/production/design/errors
- REPL: https://www.safaribooksonline.com/library/view/learning-node-2nd/9781491943113/ch04.html
- Debugging in VSCode: https://code.visualstudio.com/docs/nodejs/nodejs-debugging
- Advanced debugging with Node.js https://www.youtube.com/watch?v=_qzFJ2MPVWQ&index=8&list=PL8sJahqnzh8LOnV0s72DBt0OFBqdv9I9Y

# NODE.JS GLOBAL

**COMMAND LINE. DEBUGGING. ERRORS HANDLING**
**BY**
**ULADZIMIR DZIOMIN**