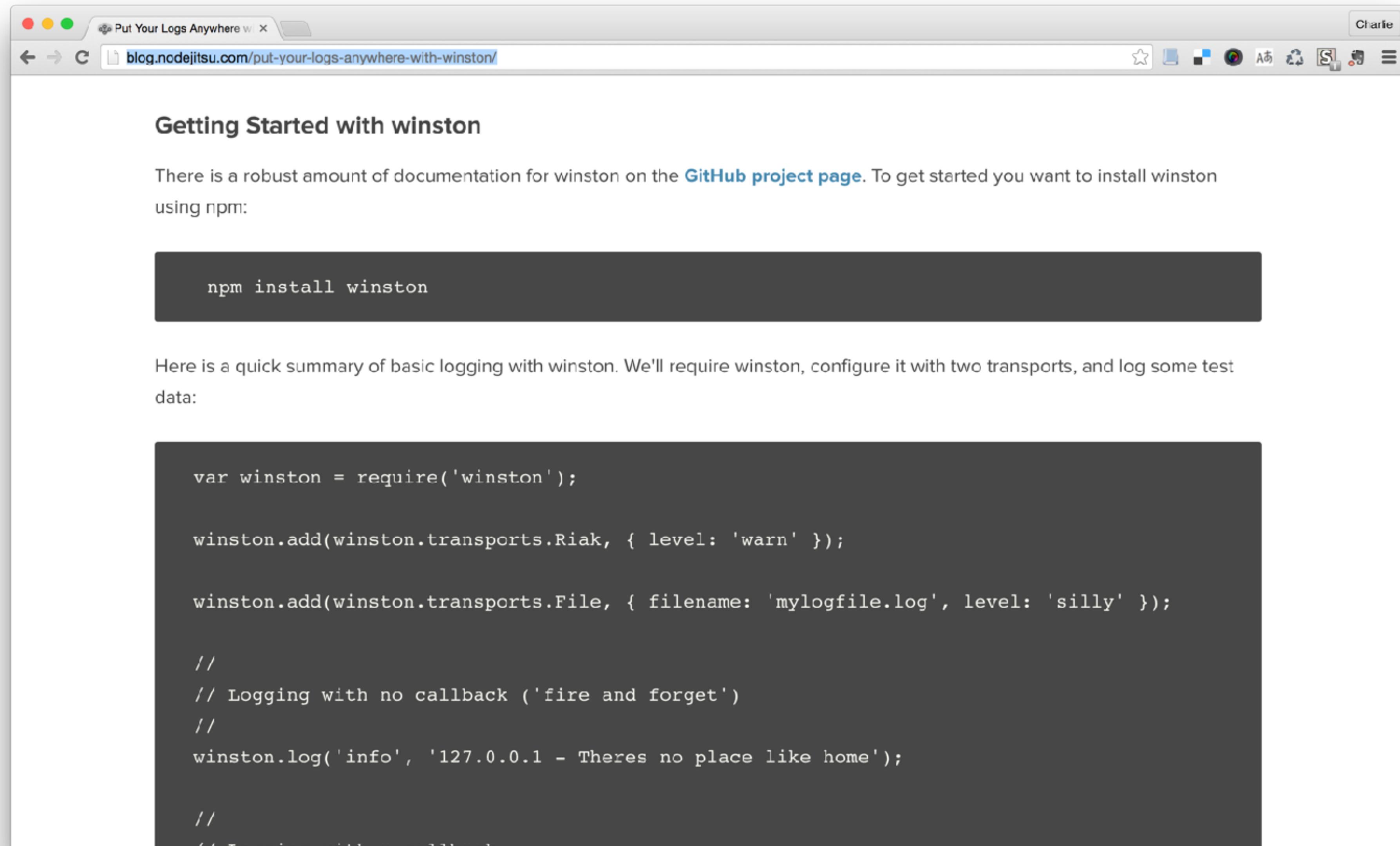




—LOOKING FOR THE—

TL;DR?



Put Your Logs Anywhere Charles

blog.nodejitsu.com/put-your-logs-anywhere-with-winston/

## Getting Started with winston

There is a robust amount of documentation for winston on the [GitHub project page](#). To get started you want to install winston using npm:

```
npm install winston
```

Here is a quick summary of basic logging with winston. We'll require winston, configure it with two transports, and log some test data:

```
var winston = require('winston');

winston.add(winston.transports.Riak, { level: 'warn' });

winston.add(winston.transports.File, { filename: 'mylogfile.log', level: 'silly' });

//
// Logging with no callback ('fire and forget')
//
winston.log('info', '127.0.0.1 - Theres no place like home');

//
// Logging with a callback

```

# WINSTON IS FROM 2010

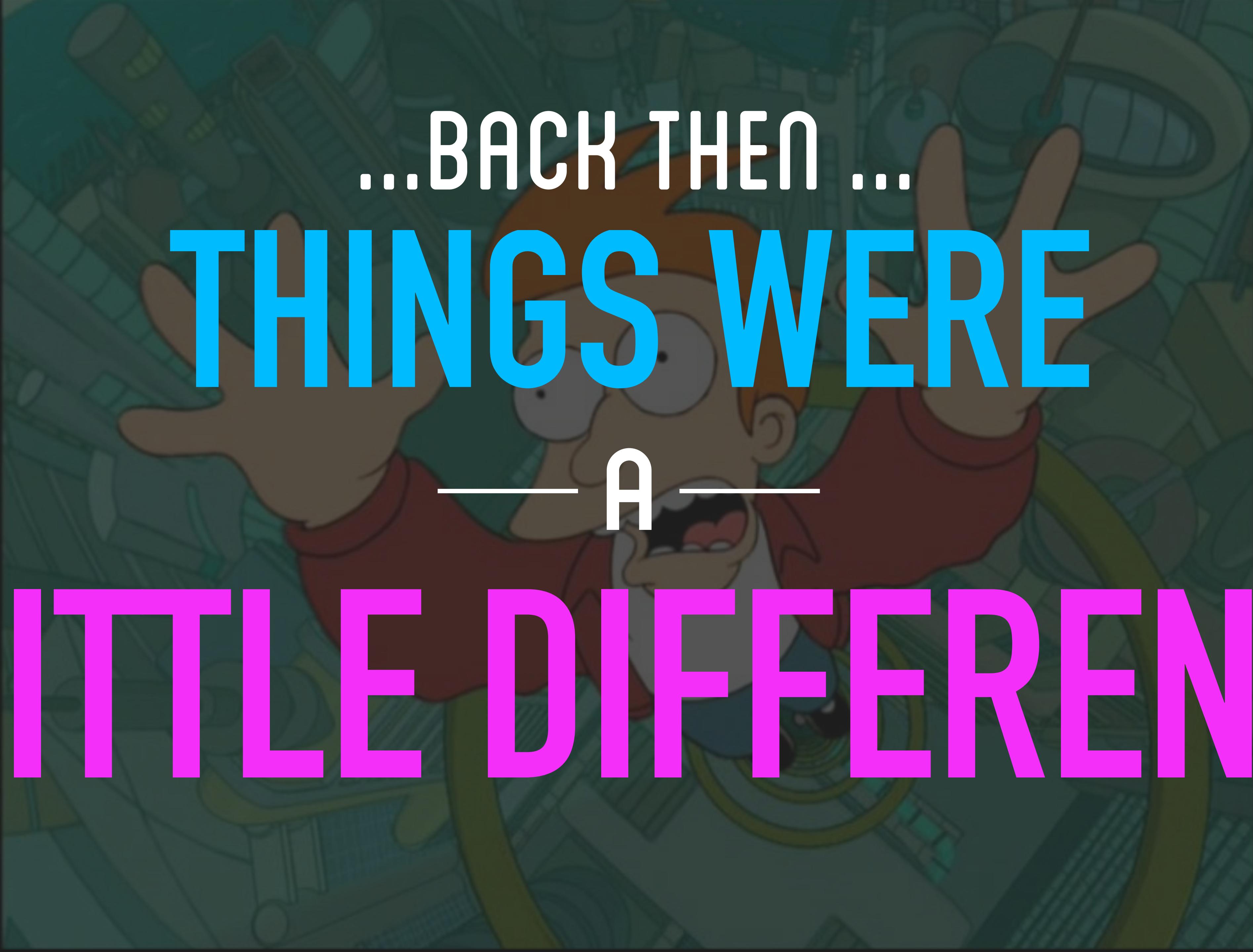


ACTUALLY  
JANUARY, 17  
2011  
BUT STARTED IN 2010



...BACK THEN ...





...BACK THEN ...

THINGS WERE

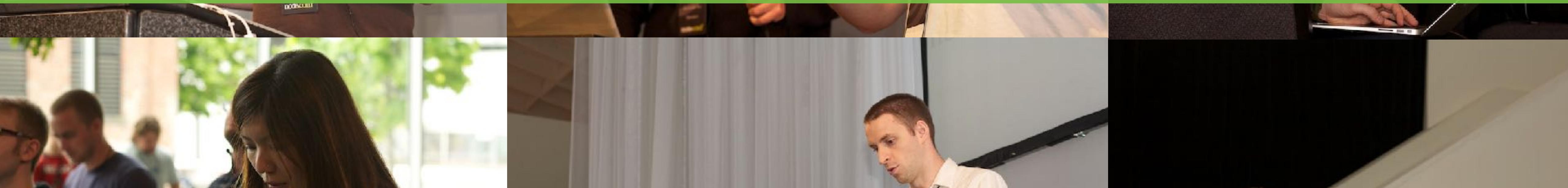
—A—

LITTLE DIFFERENT



# node.conf

NEVER BEEN ONE OF THESE



2011 BE LIKE...

# NO STREAMS API

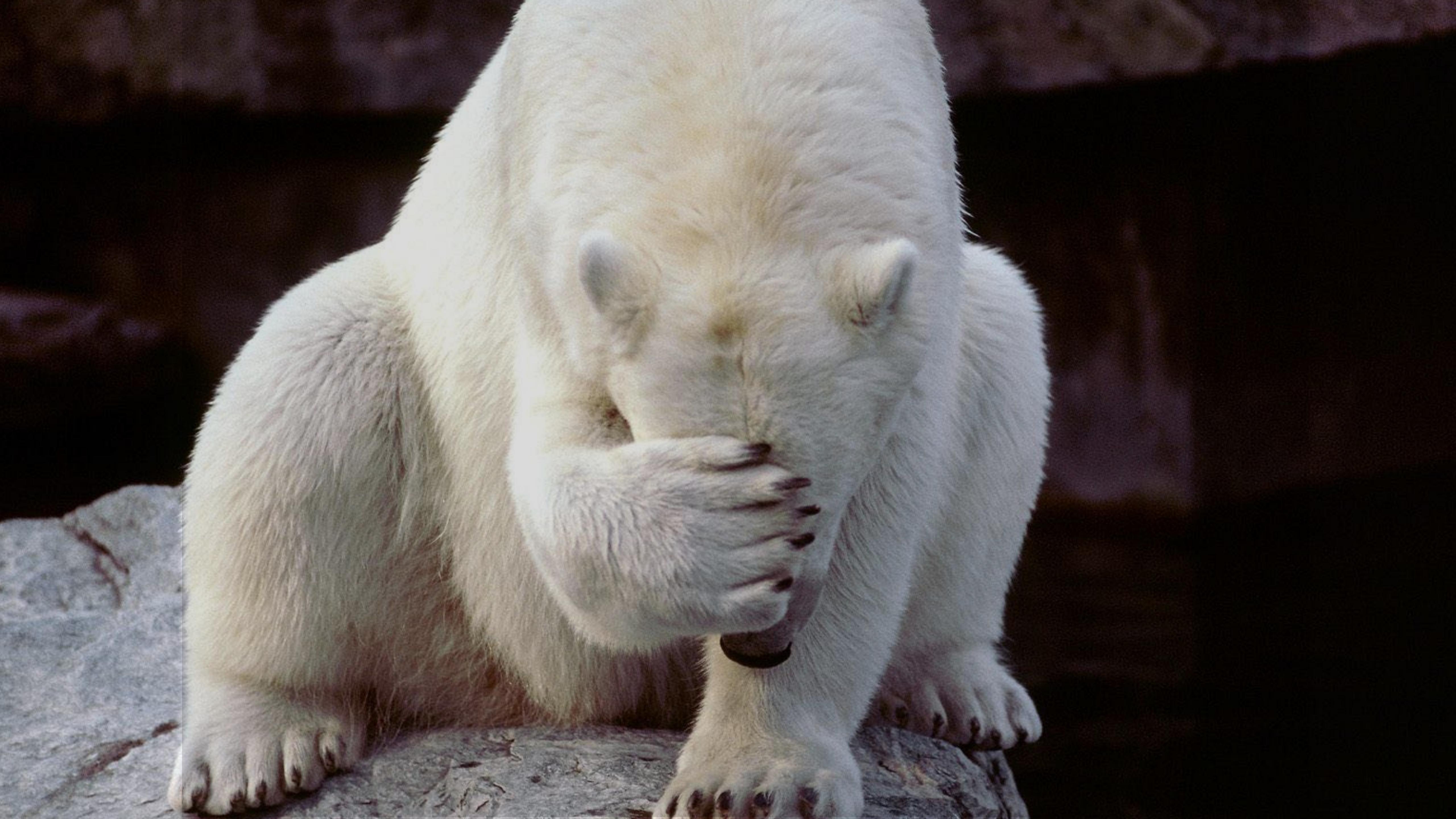


2011 BE LIKE...

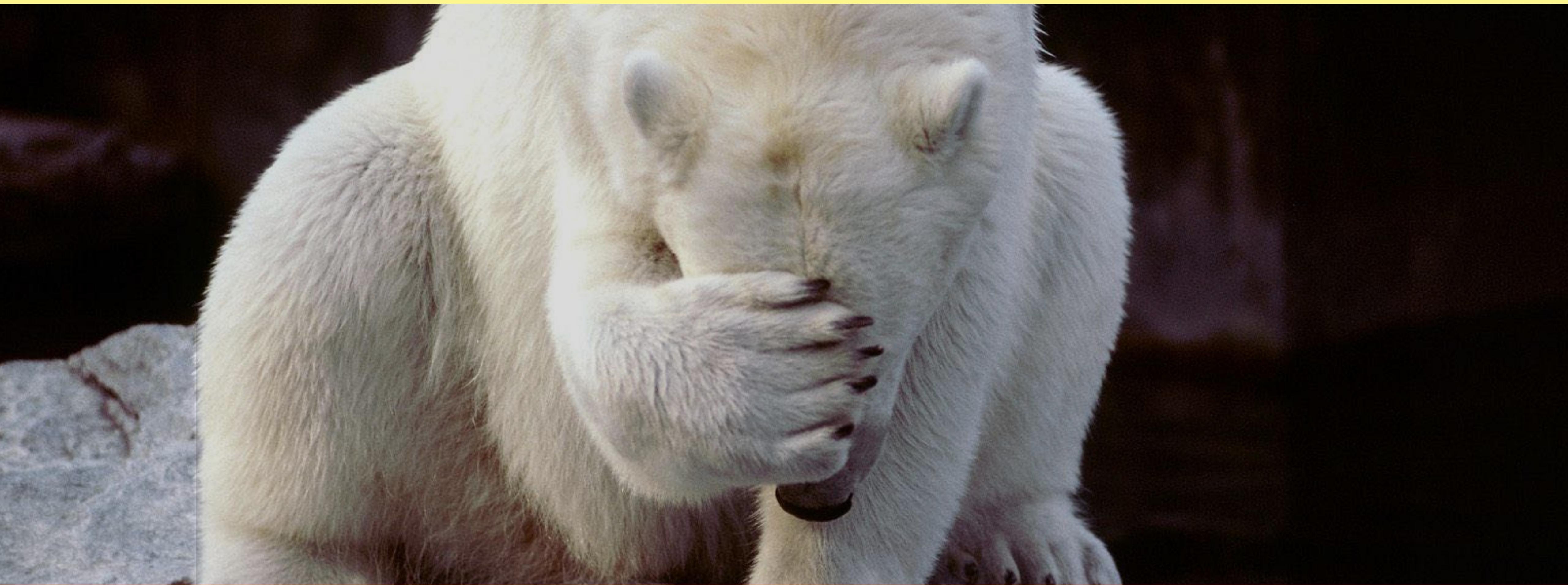
# ES5 IS STILL NEW



2011 BE LIKE...



# LIMITED CODE COVERAGE TOOLS



2011 BE LIKE...





NOT ALL  
THAT BAD THO

# BACK TO THE FUTURE



# BACK TO THE FUTURE



BY 2015 THINGS WERE MOVING ALONG...





WINSTON@1.0.0

SMALL BREAKS





WINSTON@2.0

BIGGER BREAKS





WINSTON@3.0.0

BREAK TO NEW APIS

# OPEN SOURCE





OPEN SOURCE  
IT CAN BE DONE!



IN 2015 WINSTON HAD ABOUT  
**2.6 MILLION**  
MONTHLY DOWNLOADS

BUNYAN: ~0.56M

Open Source is what  
happens when you're  
busy making other plans

- IF JOHN LENNON WROTE SOFTWARE







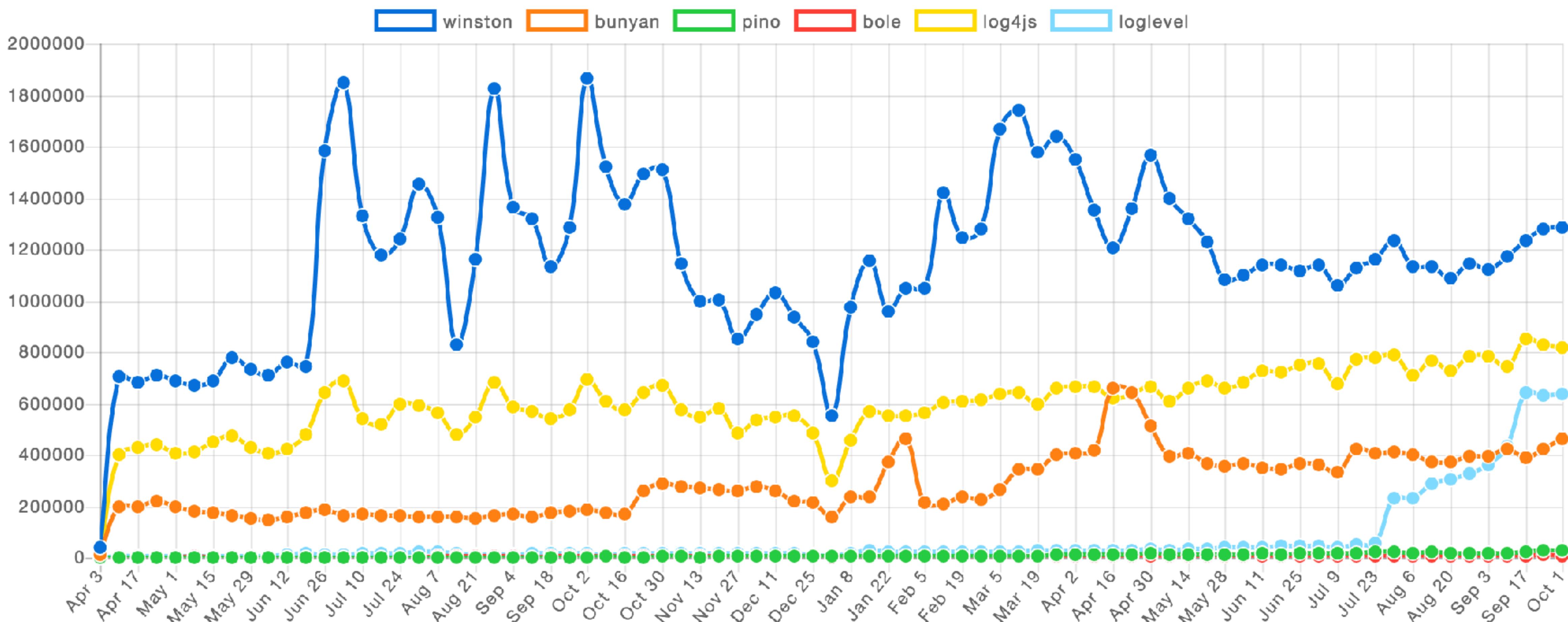
FAST FORWARD

---

TO 2017

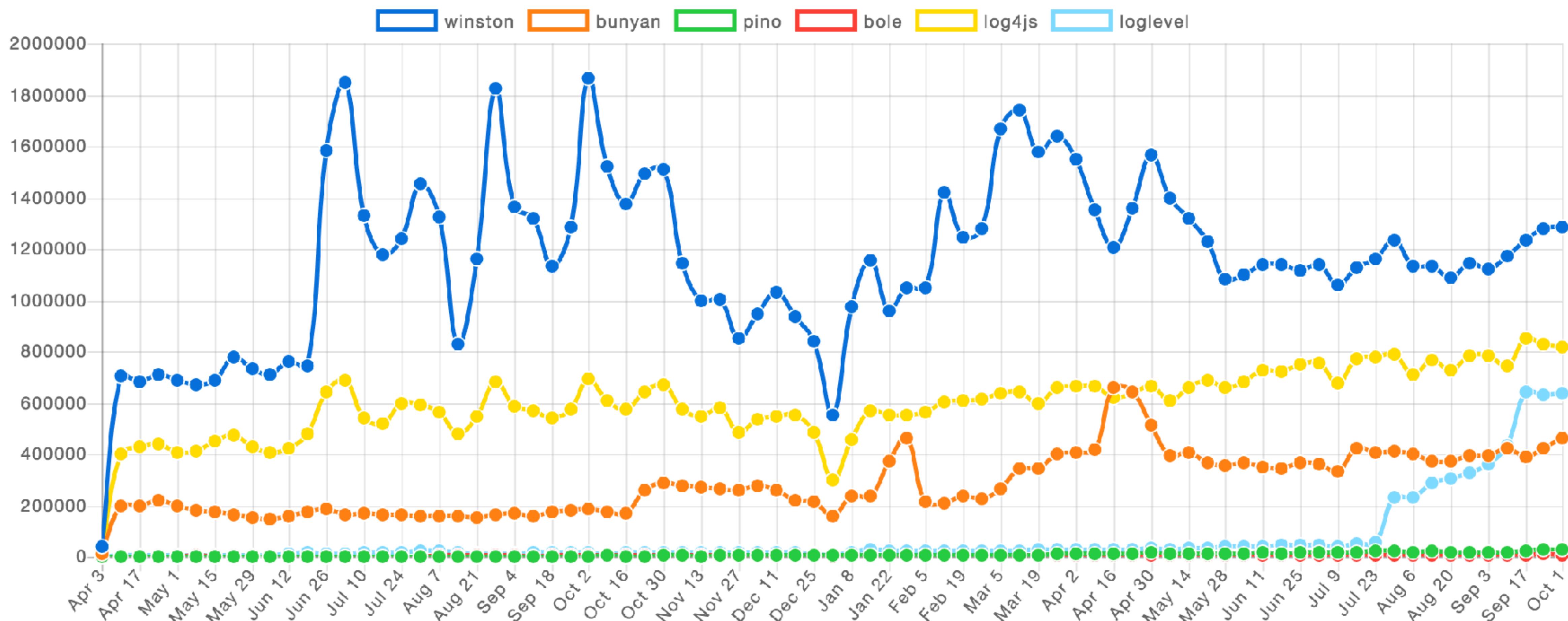
# 2017 download comparison

Downloads in past 2 Years ▾



# winston continues to “win”

Downloads in past 2 Years ▾





BUT JUST

WHY EXACTLY?



CONSIDER THE API

# SHAKESPEARE'S "LOGGING"



Autolycus 1<sup>st</sup> dress

Clown 2<sup>nd</sup> dress

Florizel 1<sup>st</sup> dress

Perdita 2<sup>nd</sup> dress

Old Shepherd 2<sup>nd</sup> dress

Bithynian Boor

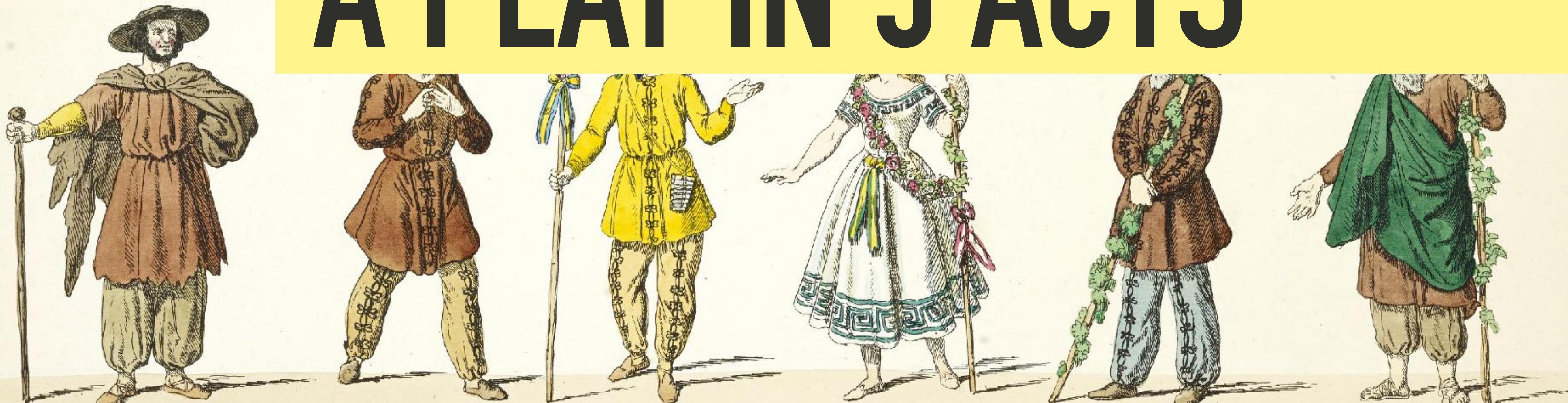
Plate 16.

Vienna: M. Trentschnsky  
Theat. Cost. 208

COSTUMES IN SHAKSPEARE'S PLAY OF  
THE WINTER'S TALE

Published by the kind permission of Charles Kean, Esq.  
London: Joseph, Myers & Co.

# SHAKESPEARE'S "LOGGING" A PLAY IN 5 ACTS



Autolycus 1<sup>st</sup> dress

Clown 2<sup>nd</sup> dress

Florizel 1<sup>st</sup> dress

Perdita 2<sup>nd</sup> dress

Old Shepherd 2<sup>nd</sup> dress

Bithynian Boor

Plate 16.

Vienna: M. Trentschnsky  
Theat. Cost. 208

COSTUMES IN SHAKSPEARE'S PLAY OF  
THE WINTER'S TALE

Published by the kind permission of Charles Kean, Esq.  
London: Joseph, Myers & Co.

— ACT ONE —

*Levels*

— ACT ONE —

*Levels*

**WHERE IN RFC5424**

**EMERGES VICTORIOUS IN THE END. ALWAYS.**



```
const syslog = {  
  emerg: 0,  
  alert: 1,  
  crit: 2,  
  error: 3,  
  warning: 4,  
  notice: 5,  
  info: 6,  
  debug: 7  
}
```



```
const npm = {  
  error: 0,  
  warn: 1,  
  http: 2,  
  info: 3,  
  verbose: 4,  
  debug: 5,  
  silly: 6  
}
```

3.  
6.

— ACT TWO —

# *Log Shipping*



— ACT TWO —

# *Log Shipping*

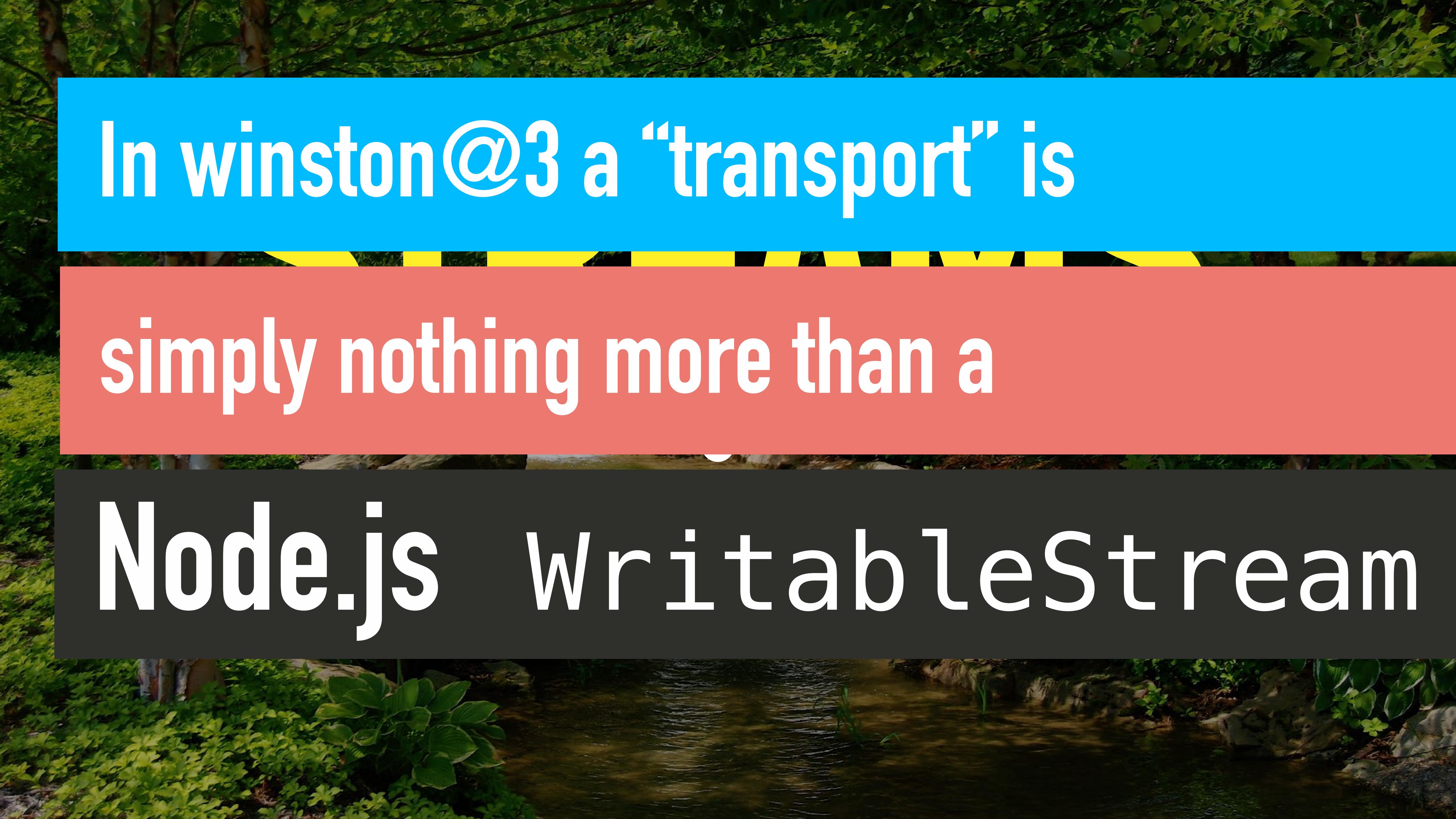
WINSTON TRANSPORTS

BUNYAN STREAMS

LOG4JS-NODE APPENDERS

A photograph of a lush green forest. In the foreground, a small stream flows over rocks, with a small waterfall visible in the middle ground. The surrounding area is filled with dense green foliage and trees. The lighting suggests a bright, sunny day.

STREAMS  
ALL THE THINGS



In `winston@3` a “transport” is

simply nothing more than a

Node.js `writableStream`

— ACT THREE —

*Querying &  
Streaming*

— ACT THREE —

*Querying &  
Streaming*

KIBANA. OR SOME SASS YOU PAY FOR.

BESPOKE ENDS TRAGICALLY – USUALLY.

— ACT FOUR —

# *Serialization & Filtering*



— ACT FOUR —

# *Serialization & Filtering*

**JSON, COLORING, PRETTYPRINTING, ALIGNMENT,**

**CUSTOM FORMATTING, TIMESTAMPS & SO SO MUCH MORE**

— ACT FIVE —

# *Defaults & Mutation*

— ACT FIVE —

# *Defaults & Mutation*

BUNYAN SERIALIZERS

MORGAN TOKENS

LOG4JS-NODE PATTERNS



— OPTIONS HELL IS —



— OPTIONS HELL IS —

AT LEAST THE  
FIFTH CIRCLE

OF

PROGRAMMING HELL





COMMON.LOG



```
//  
// ### function log (options)  
// ##### @options {Object} All information about the log serialization.  
// Generic logging function for returning timestamped strings  
// with the following options:  
  
//  
// {  
//   level:      'level to add to serialized message',  
//   message:    'message to serialize',  
//   meta:       'additional logging metadata to serialize',  
//   colorize:   false, // Colorizes output (only if `json` is false)  
//   align:      false // Align message level.  
//   timestamp:  true  // Adds a timestamp to the serialized message  
//   label:      'label to prepend the message'  
// }  
//  
exports.log = function (options) {  
  var timestampFn = typeof options.timestamp === 'function'  
    ? options.timestamp  
    : exports.timestamp,  
  timestamp    = options.timestamp ? timestampFn() : null,  
  showLevel    = options.showLevel === undefined ? true : options.showLevel,  
  meta         = options.meta !== null && options.meta !== undefined && !(options.meta instanceof Error)  
    ? exports.clone(options.meta)  
    : options.meta || null,  
  output;
```



```
//  
// ### function log (options)  
// ##### @options {Object} All information about the log serialization.  
// Generic logging function for returning timestamped strings  
// with the following options:  
//  
// {  
//   level:      'level to add to serialized message',  
//   message:    'message to serialize',  
//   meta:       'additional logging metadata to serialize',  
//   colorize:   false, // Colorizes output (only if `json` is false)  
//   align:      false // Align message level.  
//   timestamp: true  // Adds a timestamp to the serialized message  
//   label:      'label to prepend the message'  
// }  
//  
exports.log = function (options) {  
  var timestampFn = typeof options.timestamp === 'function'  
    ? options.timestamp  
    : exports.timestamp,  
  timestamp    = options.timestamp ? timestampFn() : null,  
  showLevel    = options.showLevel === undefined ? true : options.showLevel,  
  meta         = options.meta !== null && options.meta !== undefined && !(options.meta instanceof Error)  
    ? exports.clone(options.meta)  
    : options.meta || null,  
  output;
```



```
/* from: lib/winston2/transports/console.js */

var output = common.log({
  colorize:    this.colorize,
  json:        this.json,
  level:       level,
  message:     msg,
  meta:        meta,
  stringify:   this.stringify,
  timestamp:   this.timestamp,
  showLevel:   this.showLevel,
  prettyPrint: this.prettyPrint,
  raw:         this.raw,
  label:       this.label,
  logstash:    this.logstash,
  depth:       this.depth,
  formatter:   this.formatter,
  align:       this.align,
  humanReadableUnhandledException: this.humanReadableUnhandledException
});

if (this.stderrLevels[level]) {
  process.stderr.write(output + this.eol);
} else {
  process.stdout.write(output + this.eol);
}
```

# THIS IS OPTIONS HELL

```
var output = common.log({
  colorize:    this.colorize,
  json:        this.json,
  level:       level,
  message:     msg,
  meta:        meta,
  stringify:   this.stringify,
  timestamp:   this.timestamp,
  showLevel:   this.showLevel,
  prettyPrint:  this.prettyPrint,
  raw:         this.raw,
  label:        this.label,
  logstash:    this.logstash,
  depth:        this.depth,
  formatter:   this.formatter,
  align:        this.align,
  humanReadableUnhandledException: this.humanReadableUnhandledException
});
```

```
if (this.stderrLevels[level]) {
  process.stderr.write(output + this.eol);
} else {
  process.stdout.write(output + this.eol);
}
```



# INTRODUCE FORMATS

---



```
const winston = require('..');

const logger = winston.createLogger({
  transports: [new winston.transports.Console()],
  format: winston.format.combine(
    winston.format.colorize({ all: true }),
    winston.format.simple()
  )
});

logger.log('info', 'This is an information message.');
```



```
const winston = require('..');

const logger = winston.createLogger({
  transports: [new winston.transports.Console()],
  format: winston.format.combine(
    winston.format.colorize({ all: true }),
    winston.format.simple()
  )
});

logger.log('info', 'This is an information message.');
```



```
/* from: lib/winston2/transports/console.js */

var output = common.log({
  colorize:      this.colorize,
  json:         this.json,
  level:        level,
  message:      msg,
  meta:         meta,
  stringify:    this.stringify,
  timestamp:    this.timestamp,
  showLevel:    this.showLevel,
  prettyPrint:  this.prettyPrint,
  raw:          this.raw,
  label:         this.label,
  logstash:     this.logstash,
  depth:         this.depth,
  formatter:    this.formatter,
  align:         this.align,
  humanReadableUnhandledException: this.humanReadableUnhandledException
});

if (this.stderrLevels[level]) {
  process.stderr.write(output + this.eol);
} else {
  process.stdout.write(output + this.eol);
}
```



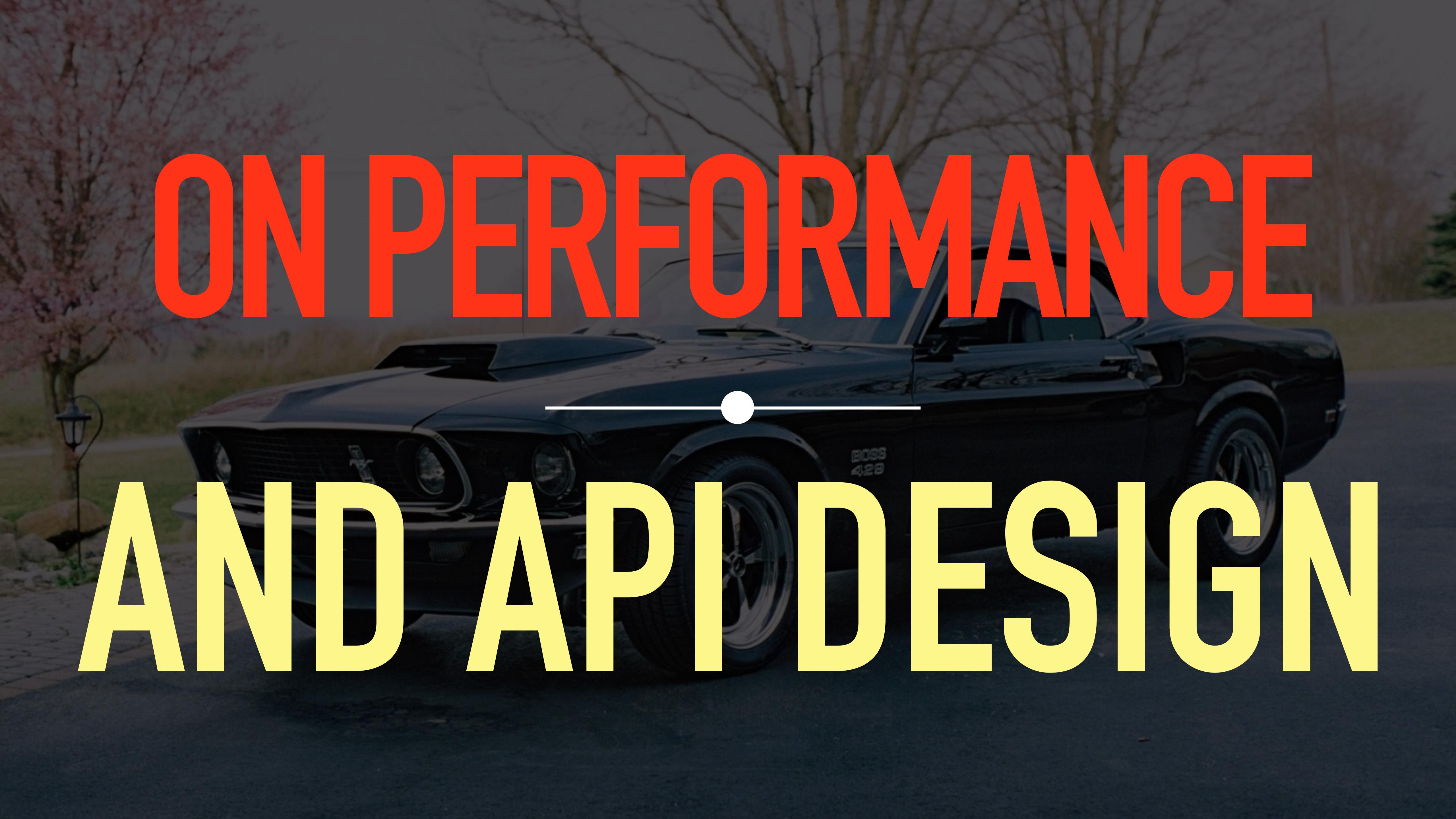
```
/* from: lib/winston3/transports/console.js */

if (this.stderrLevels[info[LEVEL]]) {
  process.stderr.write(info[MESSAGE] + this.eol);
  if (callback) { callback(); }
  return;
}

process.stdout.write(info[MESSAGE] + this.eol);
if (callback) { callback(); }
```

A LA CARTE. PAY FOR WHAT YOU USE.





# ON PERFORMANCE AND API DESIGN

---



— IT TURNS OUT —

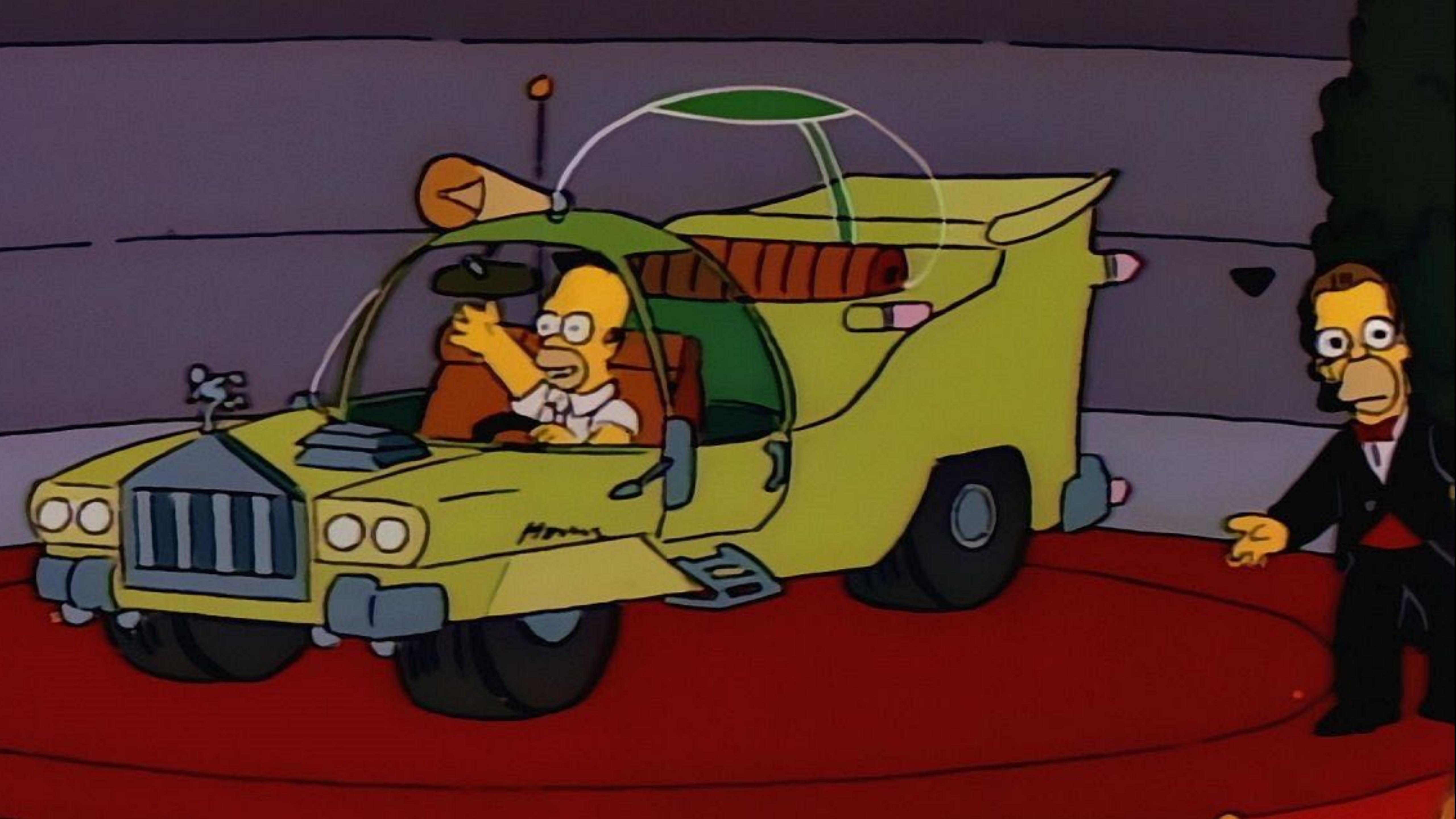


— IT TURNS OUT —

PERFORMANCE

— IS SIMPLY —

NOT EVERYTHING



— DEVELOPERS —

WANT

THE

THE HOMER CAR





— DEFINE —  
**YOUR OWN  
FORMATS**

---

---



```
/* winston.formats.colorize */

/*
 * function transform (info, opts)
 * Attempts to colorize the { level, message } of the given
 * `logform` info object.
 */
Colorizer.prototype.transform = function transform(info, opts) {
  var level = info.level;

  if (opts.level || opts.all || !opts.message) {
    info.level = this.colorize(level);
  }

  if (opts.all || opts.message) {
    info.message = this.colorize(level, info.message);
  }

  return info;
};
```



```
const { inspect } = require('util');
const { format } = require('winston');
const MESSAGE = Symbol.for('message');

/*
 * function prettyPrint (info)
 * Returns a new instance of the prettyPrint Format that "prettyPrint"
 * serializes `info` objects. This was previously exposed as
 * { prettyPrint: true } to transports in `winston < 3.0.0` .
 */
module.exports = format(function (info, opts) {
  info[MESSAGE] = inspect(info, false, opts.depth || null, opts.colorize);
  return info;
});
```



```
const { format } = require('winston');
const MESSAGE = Symbol.for('message');

/*
 * function json (info)
 * Returns a new instance of the JSON format that turns a log `info`
 * object into pure JSON. This was previously exposed as { json: true }
 * to transports in `winston < 3.0.0`.
 */
module.exports = format(function (info, opts) {
  info[MESSAGE] = JSON.stringify(info, opts.replacer || replacer, opts.space);
  return info;
});

/*
 * function replacer (key, value)
 * Handles proper stringification of Buffer output.
 */
function replacer(key, value) {
  return value instanceof Buffer
    ? value.toString('base64')
    : value;
```

A man and a woman are performing a partner stretching pose on a mat. The man is lying on his back with his legs bent and feet flat on the floor. The woman is standing over him, holding his feet and stretching his legs. They are in a well-lit room with a large plant in the background.

— NOT AS —  
**EASY AS**  
— IT LOOKS —

## What is the Rosetta Stone?

The Rosetta Stone carries an inscription in different languages which helped decipher the ancient Egyptian hieroglyphic script. It is the only surviving fragment of a larger stone slab (stela) recording a decree on 27 March, 196 BC.

At the top the decree was written in hieroglyphs, the traditional script of Egyptian monuments. Then, nearly 3000 years later, in the middle, the same decree was written in Demotic, the everyday script of literate Egyptians, and at the bottom in Greek, the language used by the government.

At this time Egypt was ruled by a Greek dynasty and the decree was issued in honour of the new king Ptolemy V Epiphanes. It records the decision of the Egyptian priests to send him a stela built in honour for Ptolemy's concession to the Egyptian temples. The granite stela was placed in a temple complex in the city of Sais near Rhoda (Rosetta).

## Hieroglyphs

## The key to Egyptian hieroglyphs

As soon as the Rosetta Stone was discovered, scholars realised that it might help decipher the mysterious Egyptian hieroglyphs, since the Greek inscription, which could be read, stated that each script on the Stone recorded the same decree.

In England and France two exceptional men were working on hieroglyphs: Thomas Young (1773–1829) and Jean-François Champollion (1790–1832). Earlier scholars had already guessed that signs, or cartouches, in hieroglyphic inscriptions probably enclosed royal names. Young used the cartouches on the Rosetta Stone to work out that some hieroglyphs wrote the names of the Greek royal names Ptolemy, but he thought most hieroglyphs were symbolic images.

Cartouche (written from right to left):



= ΠΤΟΛΕΜΑΙΟΣ  
of Ptolemy

## Demotic

## Demotic script

Stylized script:

# — ES6 SYMBOLS —

# SYMBOLS!

— ES6 SYMBOLS —

# EVERWHERE



# —ES6 SYMBOLS—

```
const LEVEL = Symbol.for('level');      // Treated as immutable for priorities
const MESSAGE = Symbol.for('message'); // Final serialized output
```



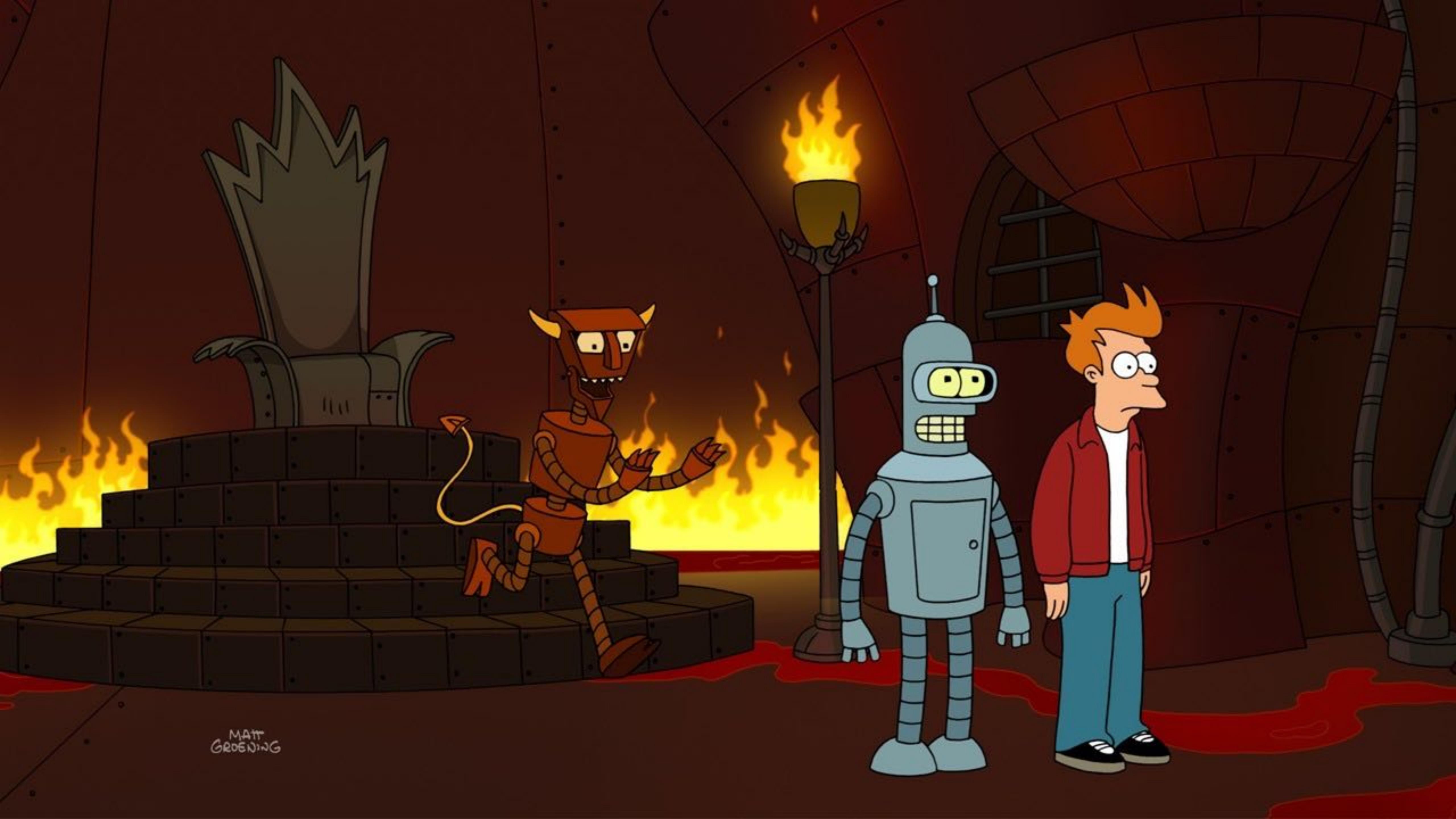
# —ES6 SYMBOLS—

```
const LEVEL = Symbol.for('level');      // Treated as immutable for priorities
const MESSAGE = Symbol.for('message'); // Final serialized output
```

LESS MEMORY OVERHEAD

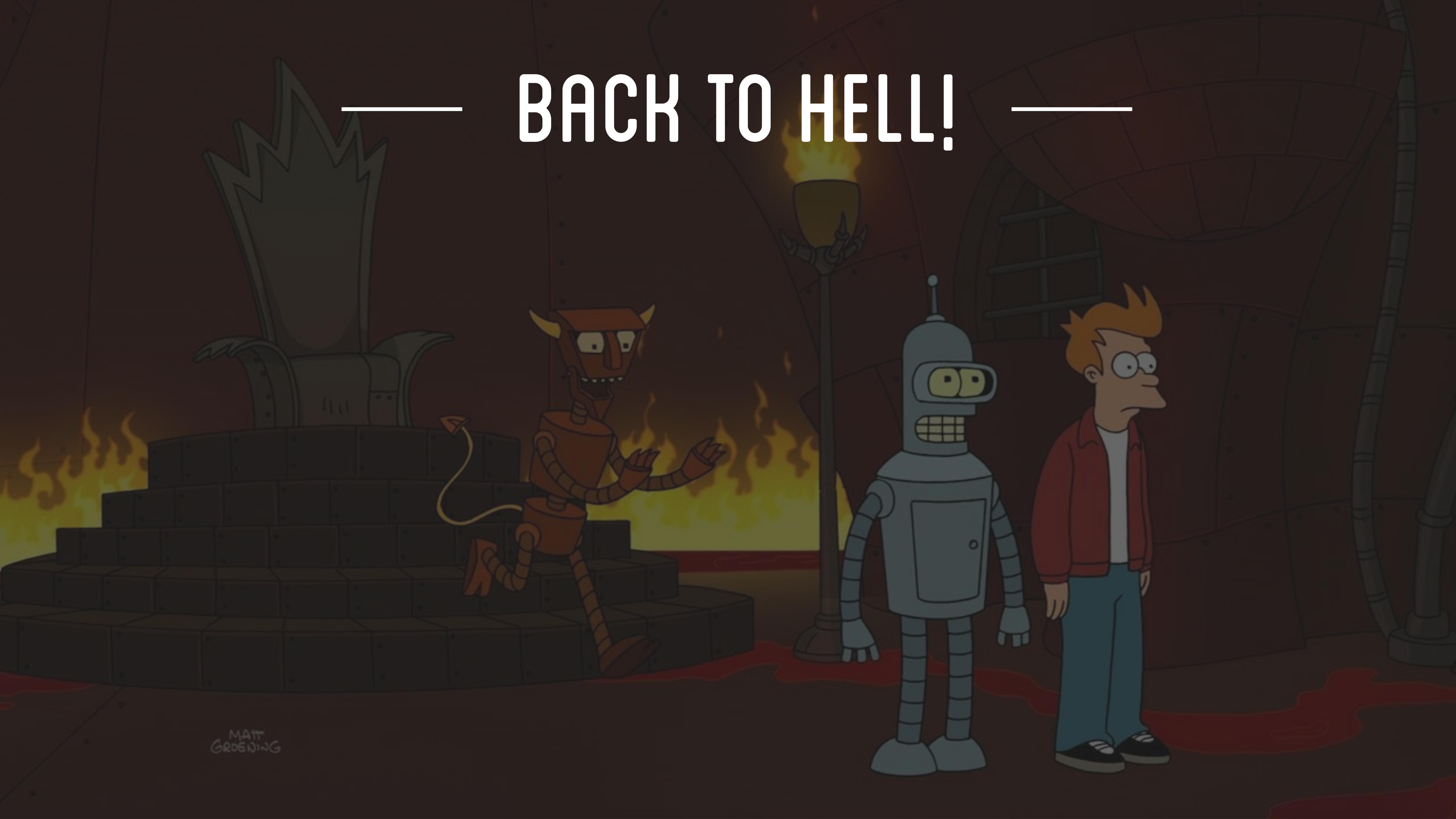
NON-ENUMERABLE / SEMI-PRIVATE

WILL NOT APPEAR IN DEFAULT JSON.STRINGIFY



MATT  
GROENING

# — BACK TO HELL! —



MATT  
GROENING

— BACK TO HELL! —

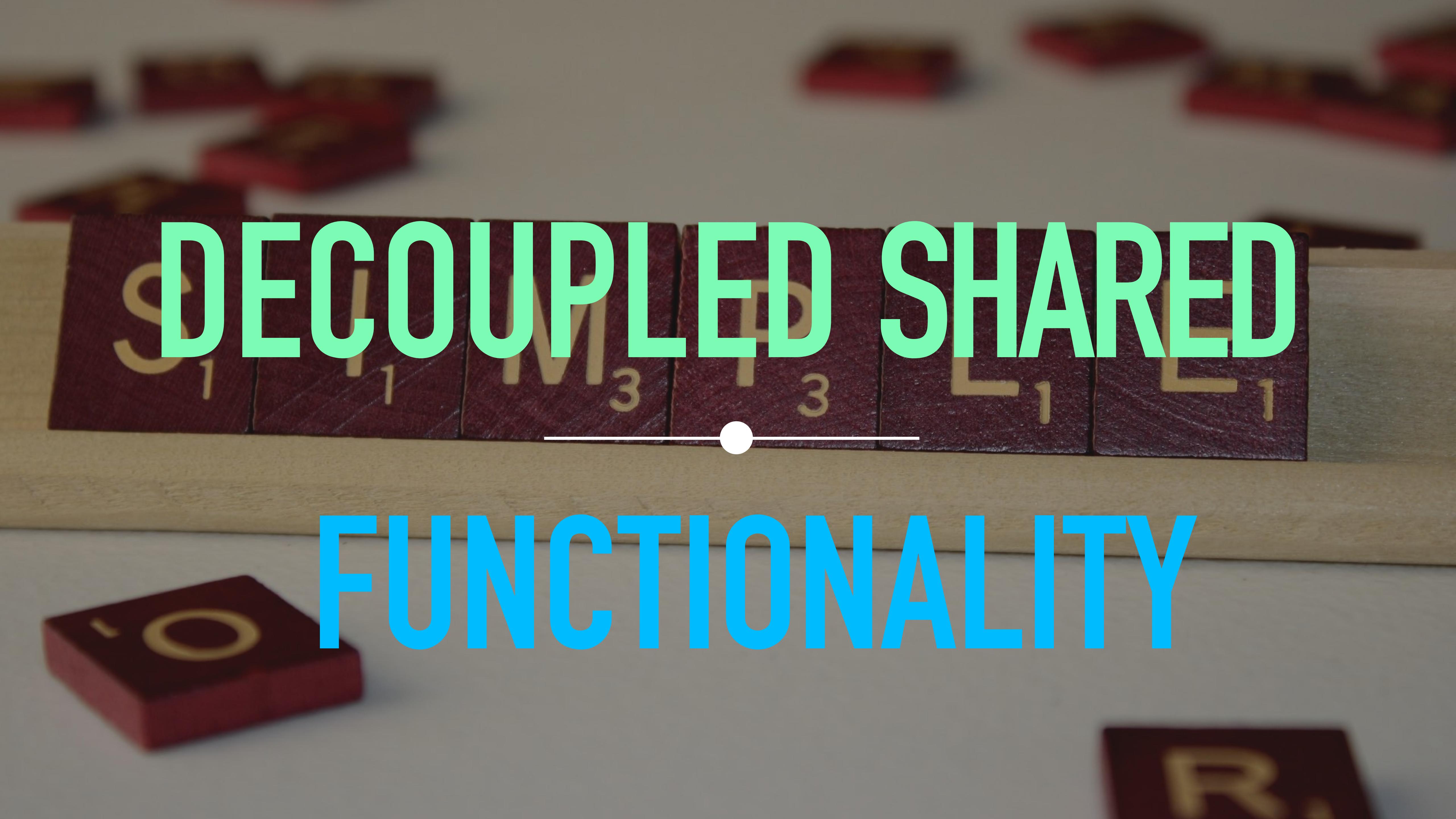
**COUPLING IS  
THE NINTH CIRCLE  
OF  
PROGRAMMING HELL**



S<sub>1</sub> I<sub>1</sub> M<sub>3</sub> P<sub>3</sub> L<sub>1</sub> E<sub>1</sub>

. O

R



DECOUPLED SHARED

FUNCTIONALITY

REQUIERED MODULES

```
require('winston-transport');
```

```
require('abstract-winston-transport');
```

FUNCTIONALITY

```
● ● ●

const Transport = require('winston-transport');

//  
// Inherit from `winston-transport` so you can take advantage  
// of the base functionality and `^.exceptions.handle()`.  
//  
module.exports = class YourCustomTransport extends Transport {  
  constructor(opts) {  
    super(opts);  
    //  
    // Consume any custom options here. e.g.:  
    // - Connection information for databases  
    // - Authentication information for APIs (e.g. loggly, papertrail,  
    //   logentries, etc.).  
    //  
  }  
  
  log(info, callback) {  
    setImmediate(function () {  
      this.emit('logged', info);  
    });  
  
    // Perform the writing to the remote service  
    callback();  
  }  
};
```



```
/**  
 * winston.format and winston.addColors are  
 * a part of logform. All of winston's formatting  
 * is decoupled from winston itself.  
 */  
const { format, levels } = require('logform');  
winston.format = format;  
winston.addColors = levels;
```



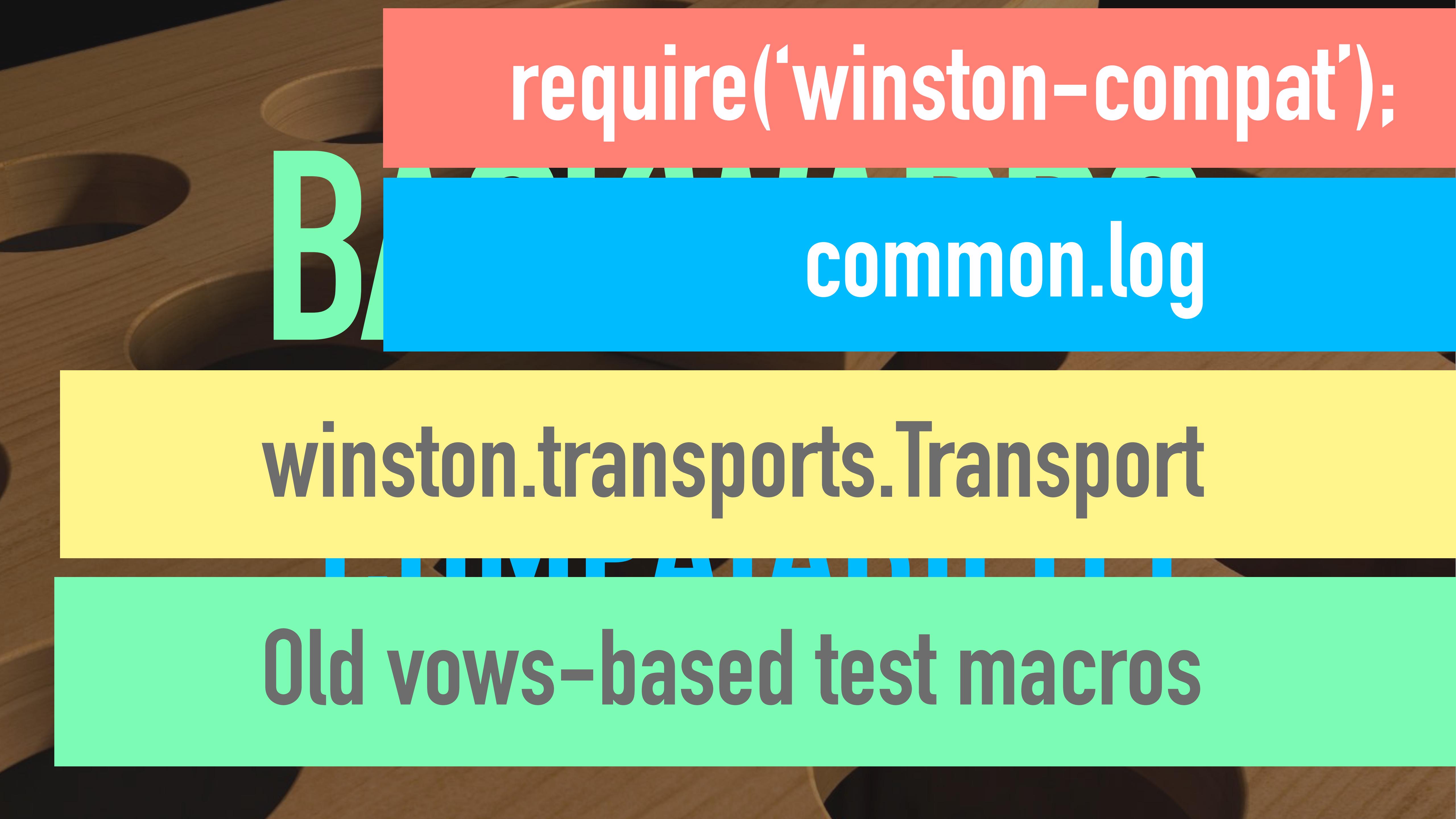
# require('logform')



# BACKWARDS

---

# COMPATABILITY



```
require('winston-compat');
```

**Bi** common.log

winston.transports.Transport

Old vows-based test macros

# BENCHMARKS SAY UP TO...



BENCHMARKS SAY UP TO...

375% FASTER

THAN WINSTON@2

FASTER THAN BUNYAN



— ACTIVELY SEEKING —

**NEW MAINTAINERS**

— ACTIVELY SEEKING —

# NEW MAINTAINERS

YES! THAT MEANS YOU!

SHOULD JOIN WINSTON

FIN



npm install winston@next