ASYNCHRONOUS ZERO

WHO AM I

all my entire life ... I do something with JavaScript

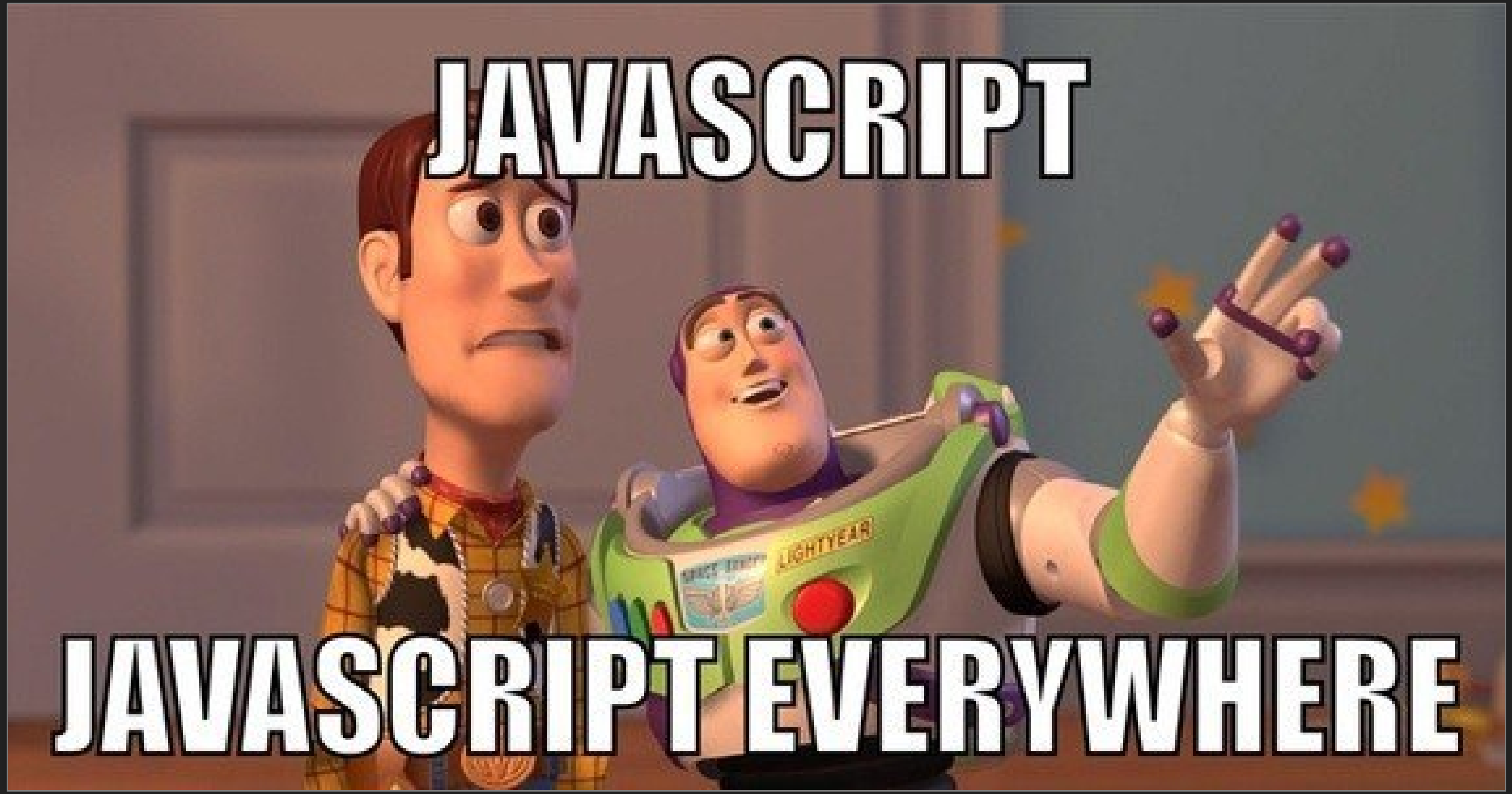
nowdays for ukit.group

1st commercial site with JS at 2001.
Then MS driven JS: IIS Active Server Pages,
Windows Scripting Host for Active Directory &
HTML Applications for Intranet.
Then Rhino & Nashorn for Alfresco & Asterisk.
Then XUL Runner. Cordova. ExtJS.

AND FINALLY NODEJS.

meanwile former researcher of IT Economy
... and still very interested

TEAM MASCOTS ME



— I HAVE A DREAM —

is really hard to think of an idea for 15 years

dreams are the sort of things you
can't stop think about & get rid of

SO THIS TALK WAS MADE

- seems important ... [for me]
- reveal current technology state
- tell about problems & solutions

… INTRO …

-1

IN THE BEGINNING WAS THE WORD

AND WORD WAS

JavaScript

single threaded synchronous programming language

BUT THEN THEY INVENTED

ASYNCHRONY

```
// and continued reproducing it everywhere
```

```
// for everything they do
```

```
// & for bugs
```

AND THIS

HOW ALL BECOMDE

TOO COMPLICATED

```
// and going only worse
```

RELATIONS

&

CONSIDERATIONS

most of all significant data
collected from issues or meetings of

Diagnostics Working Group

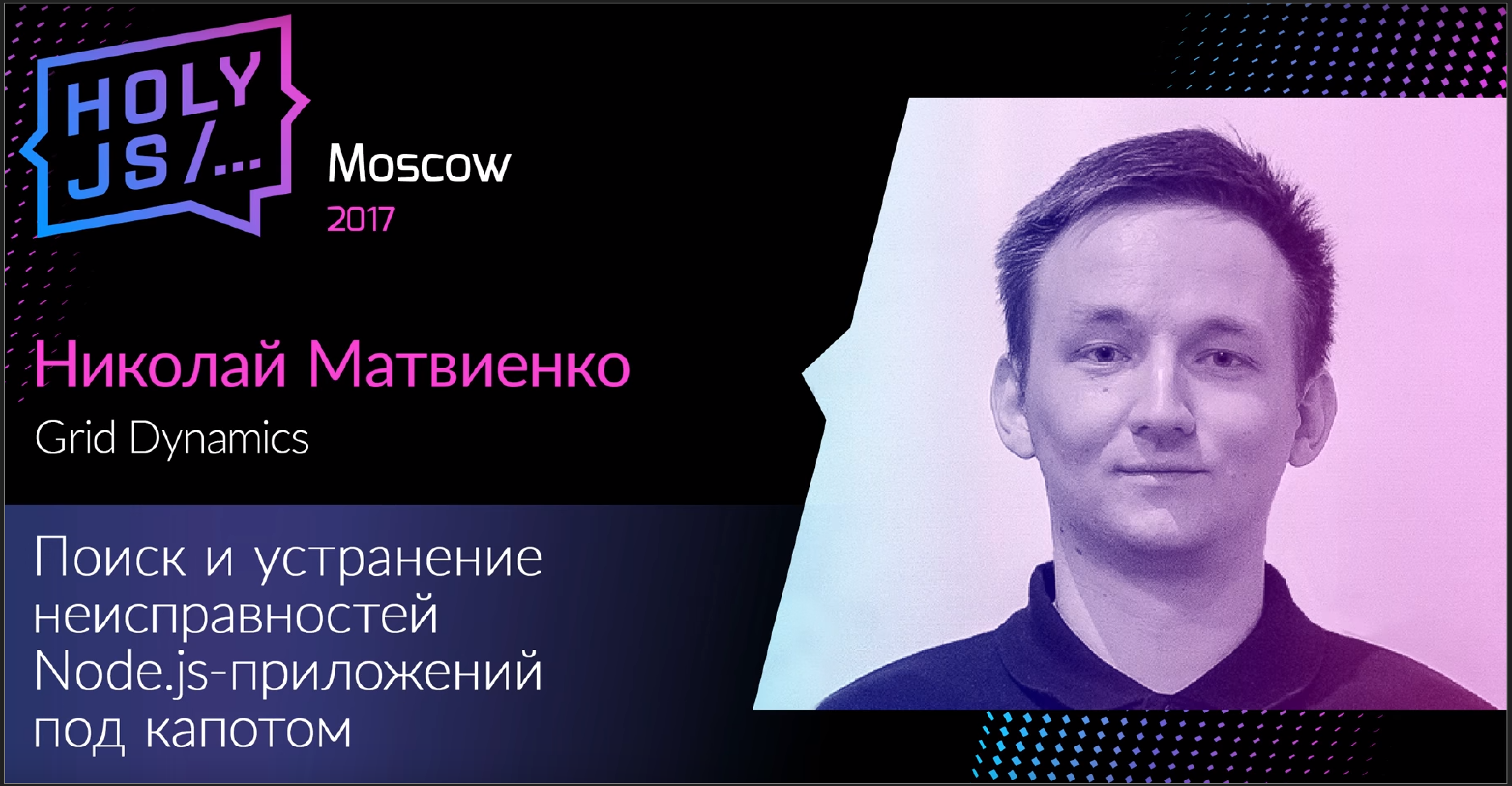
The goal of this WG is to ensure Node provides a set of comprehensive, documented, extensible diagnostic protocols, formats, and APIs to enable tool vendors to provide reliable diagnostic tools for Node.

Work is divided into several domains:

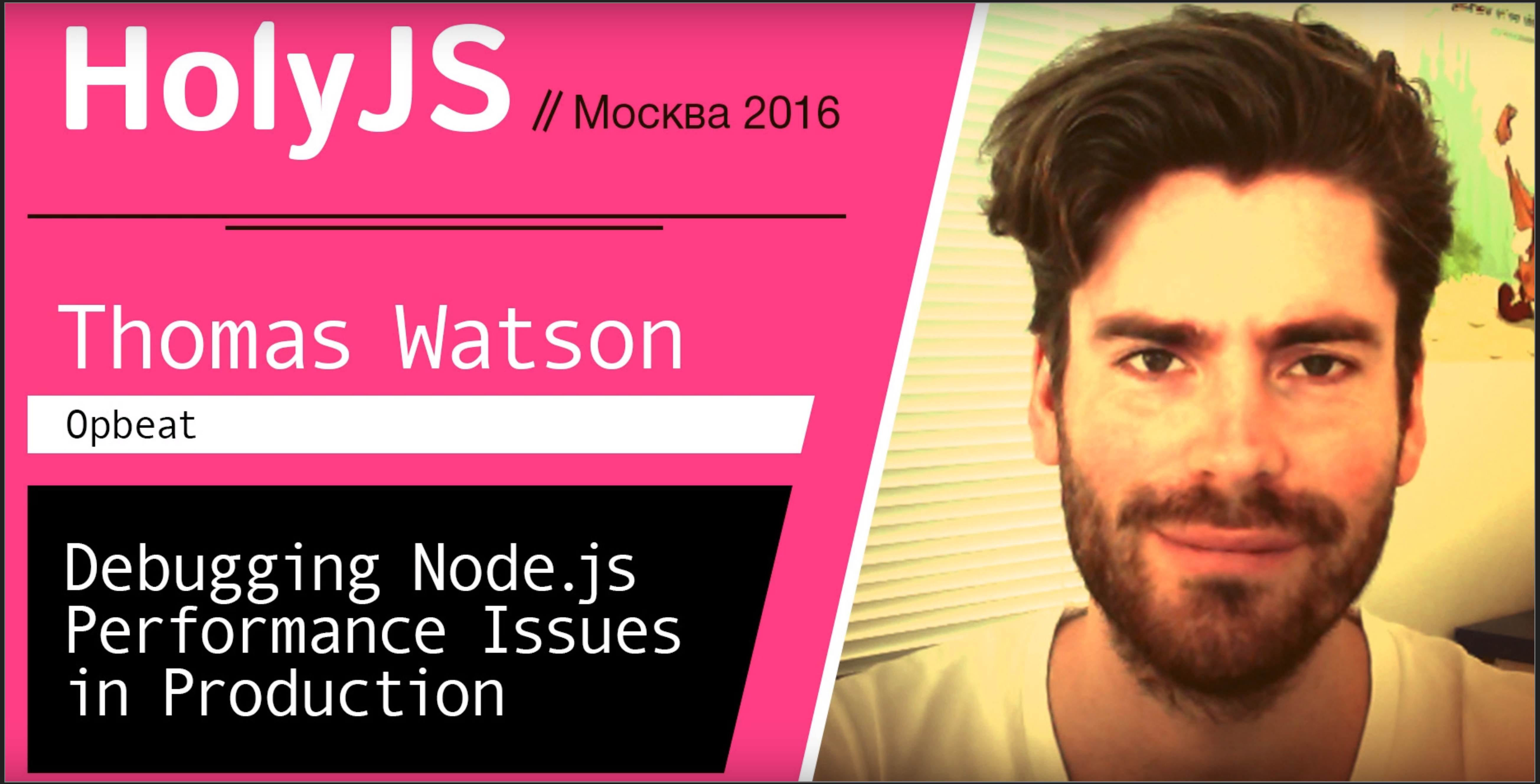
- Tracing
- Profiling
- Heap and Memory Analysis
- Step Debugging

Background, reference documentation, samples, and discussion for each domain is contained within its folder.

DEBUGGING IN PRODUCTION



DEBUGGING IN PRODUCTION



NODE FATAL FLOW

WHAT WE HAVE

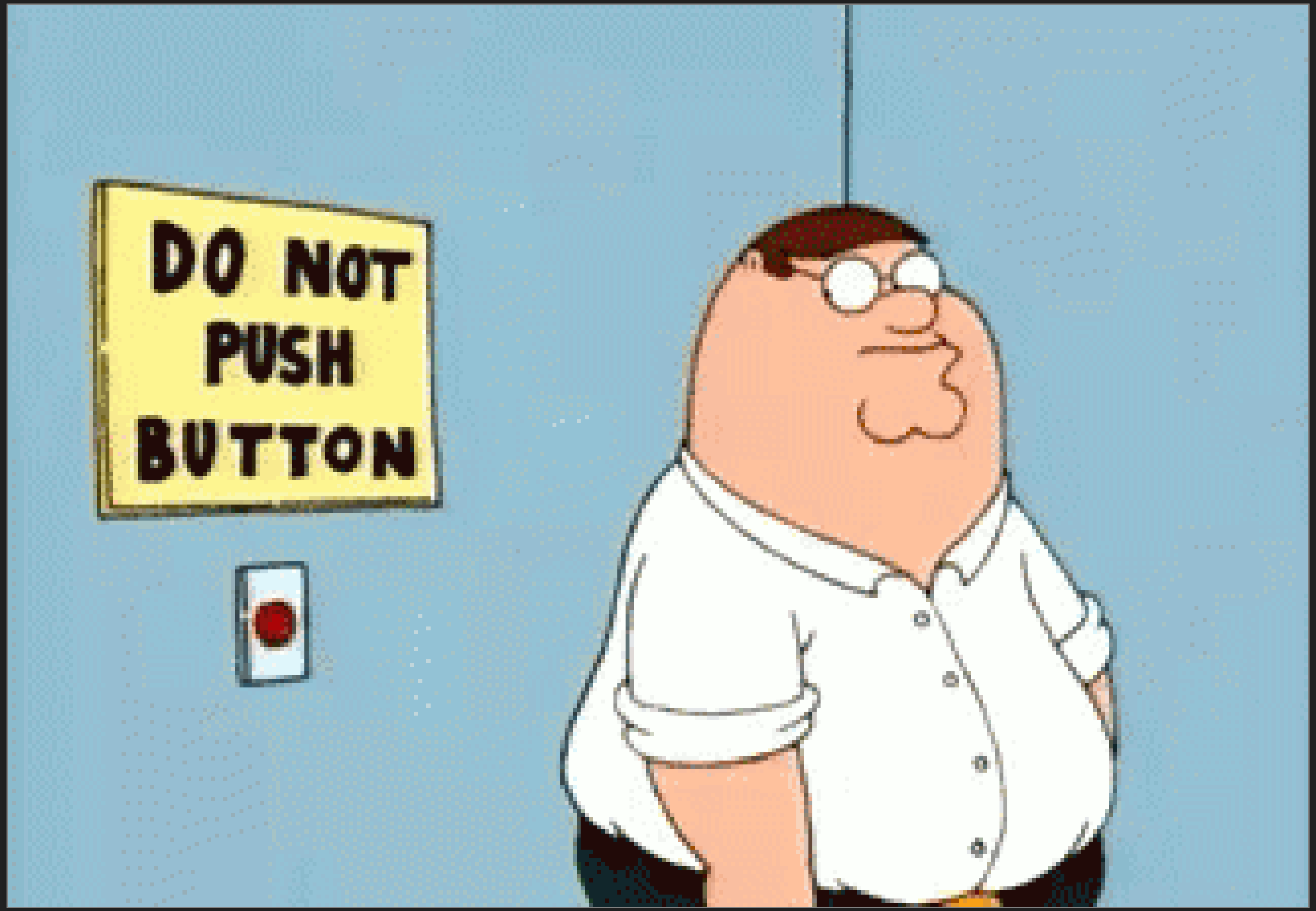
an approach to be meaningfull
[w]~[o] statless condition to any key

metaphor

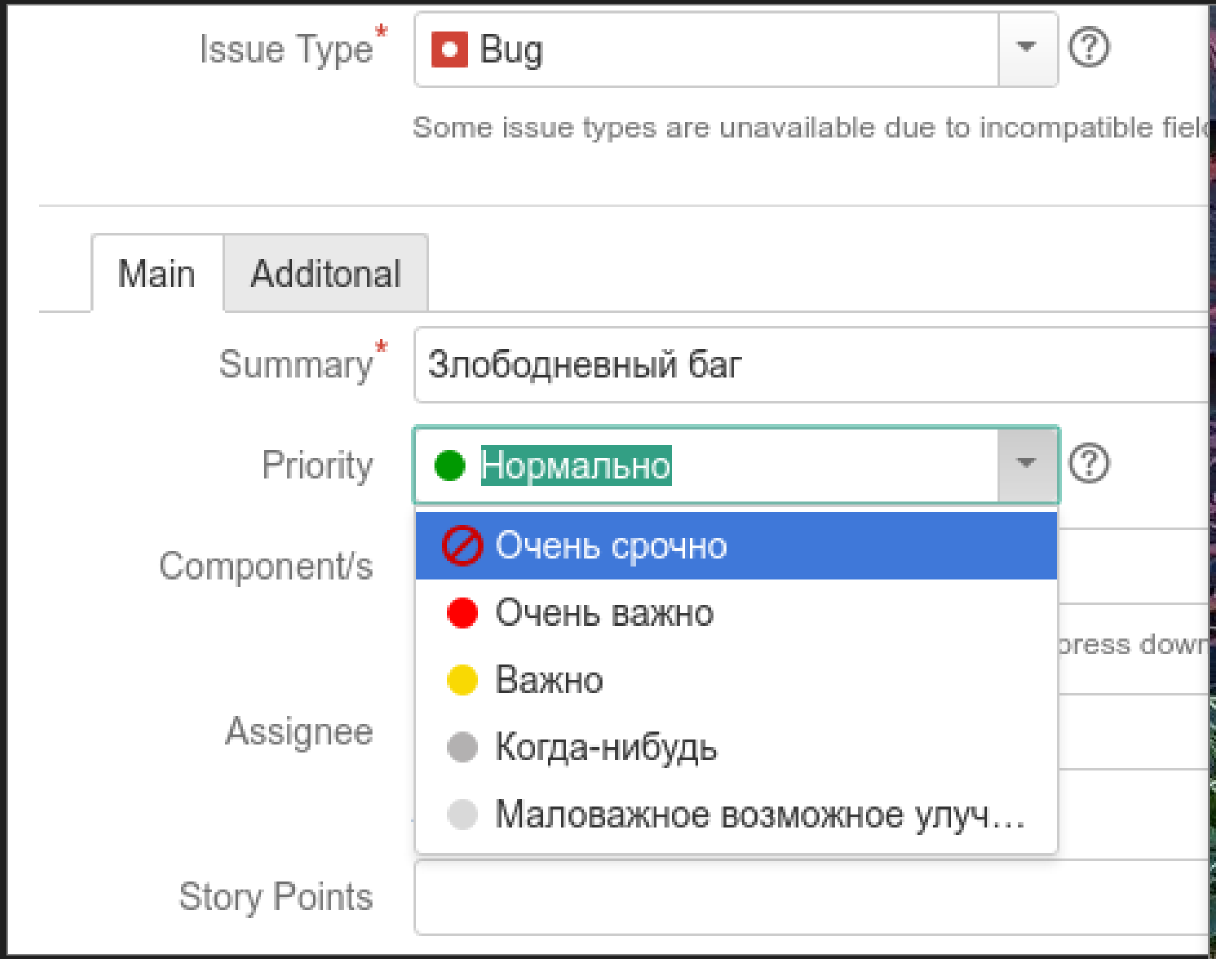
OF ASYNC

4 CODING

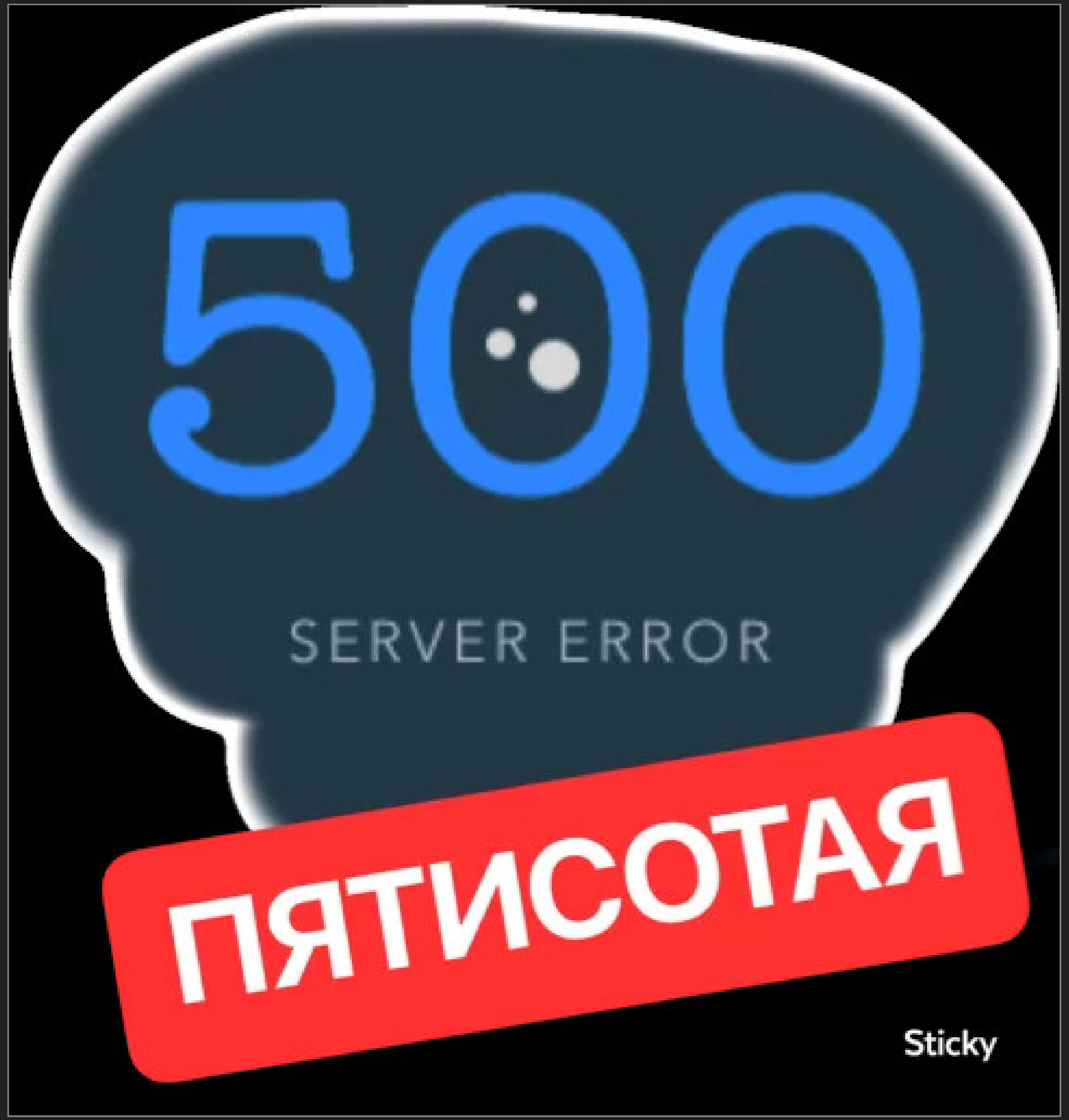
— AND SOMEHOW ... SOMEWHERE —



SOMEONE ASSIGNS ME A BUG

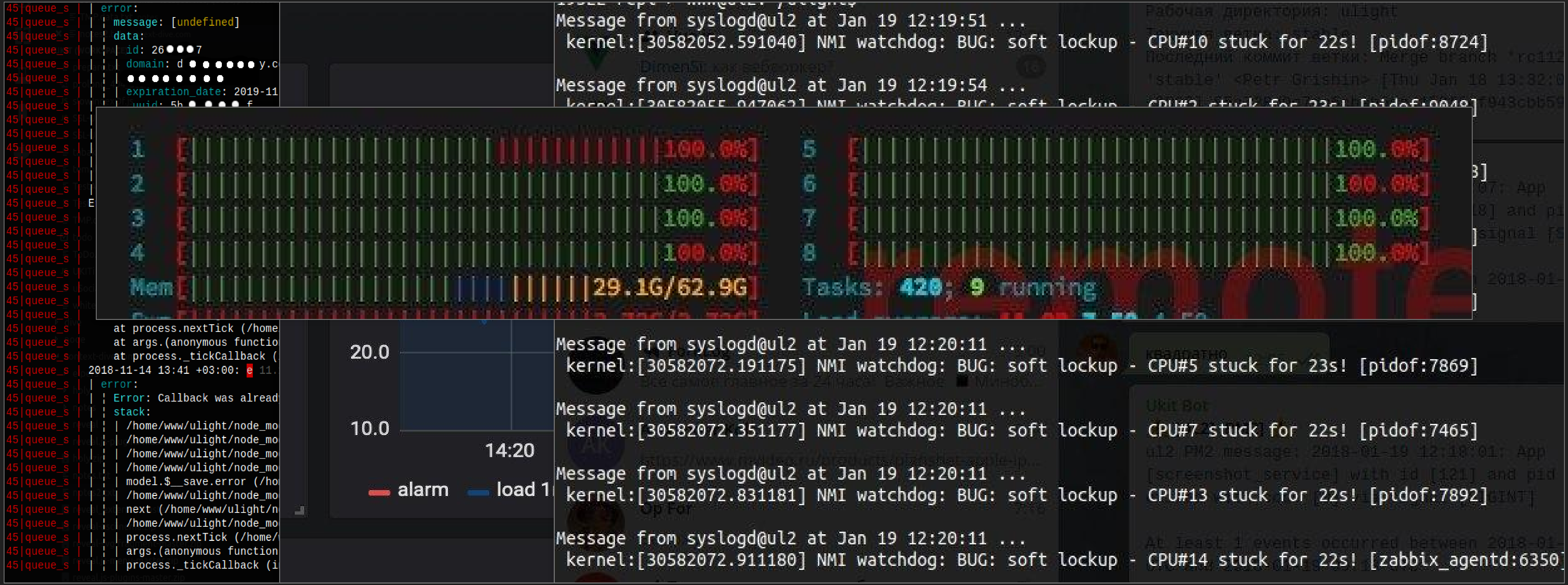


and it that situation



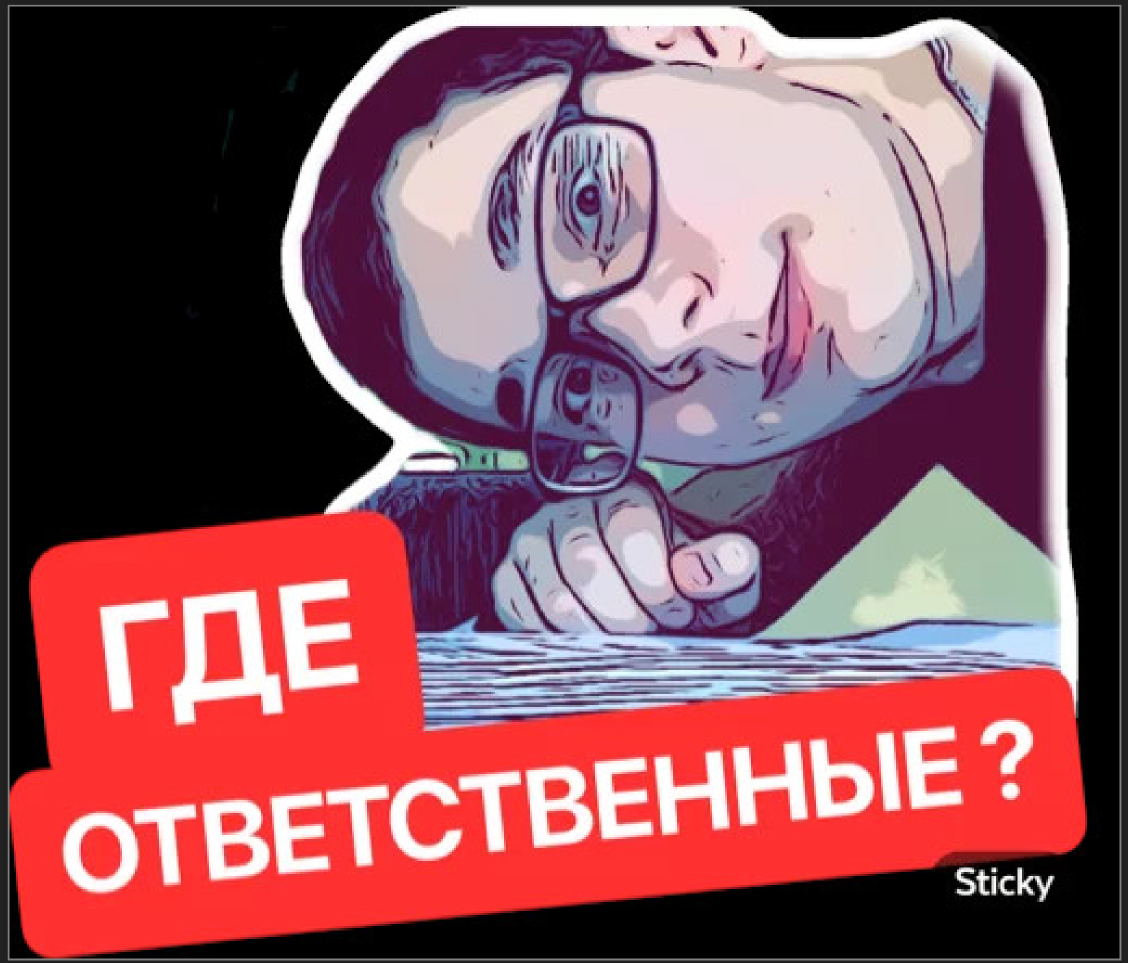
```
45|queue_s | | error:
45|queue_s | | ¦ message: [undefined]
45|queue_s | | ¦ data:
45|queue_s | | ¦ | id: 26[redacted]7
45|queue_s | | ¦ | domain: d[redacted]y.com
45|queue_s | | ¦ | [redacted]
45|queue_s | | ¦ | expiration_date: 2019-11-14T10:41:19Z
45|queue_s | | ¦ | _uuid: 5b[redacted]f
45|queue_s | | userDomainName: d[redacted]y.com
45|queue_s | | data:
45|queue_s | | ¦ checkDomainIsPaid: [null]
45|queue_s | | ¦ checkDomainContact: [null]
45|queue_s | | ¦ checkDomainContactReg: [null]
45|queue_s | | ¦ makeRegistration: [null]
45|queue_s | Error: Callback was already called.
45|queue_s |     at /home/www/ulight/node_modules/async/lib/async.js:43:36
45|queue_s |     at /home/www/ulight/node_modules/async/lib/async.js:694:17
45|queue_s |     at /home/www/ulight/node_modules/async/lib/async.js:173:37
45|queue_s |     at /home/www/ulight/node_modules/mongoose/lib/model.js:4506:16
45|queue_s |     at model.$__save.error (/home/www/ulight/node_modules/mongoose/lib/model.js:401:7)
45|queue_s |     at /home/www/ulight/node_modules/kareem/index.js:315:21
45|queue_s |     at next (/home/www/ulight/node_modules/kareem/index.js:209:27)
45|queue_s |     at /home/www/ulight/node_modules/kareem/index.js:182:9
45|queue_s |     at process.nextTick (/home/www/ulight/node_modules/kareem/index.js:452:38)
45|queue_s |     at args.(anonymous function) (/home/www/.nvm/versions/node/v10.13.0/lib/node_modules/pm2/node_modules/event-loop-inspector/index.js:138:29)
45|queue_s |     at process._tickCallback (internal/process/next_tick.js:61:11)
45|queue_s | 2018-11-14 13:41 +03:00: e 11.14 13:41:26 service/queue.js:69:8 QueueService | uncaughtException
45|queue_s | | error:
45|queue_s | | ¦ Error: Callback was already called.
45|queue_s | | ¦ stack:
45|queue_s | | ¦ | /home/www/ulight/node_modules/async/lib/async.js:43:36
45|queue_s | | ¦ | /home/www/ulight/node_modules/async/lib/async.js:694:17
45|queue_s | | ¦ | /home/www/ulight/node_modules/async/lib/async.js:173:37
45|queue_s | | ¦ | /home/www/ulight/node_modules/mongoose/lib/model.js:4506:16
45|queue_s | | ¦ | model.$__save.error (/home/www/ulight/node_modules/mongoose/lib/model.js:401:7)
45|queue_s | | ¦ | /home/www/ulight/node_modules/kareem/index.js:315:21
45|queue_s | | ¦ | next (/home/www/ulight/node_modules/kareem/index.js:209:27)
45|queue_s | | ¦ | /home/www/ulight/node_modules/kareem/index.js:182:9
45|queue_s | | ¦ | process.nextTick (/home/www/ulight/node_modules/kareem/index.js:452:38)
45|queue_s | | ¦ | args.(anonymous function) (/home/www/.nvm/versions/node/v10.13.0/lib/node_modules/pm2/node_modules/event-loop-inspector/index.js:138:29)
45|queue_s | | ¦ | process._tickCallback (internal/process/next_tick.js:61:11)
```

WHEN YOU HAVE STACK... BUT

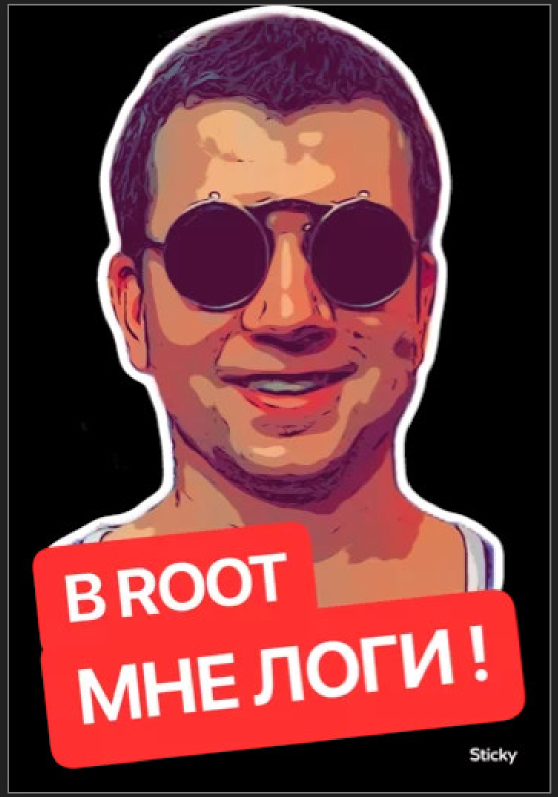


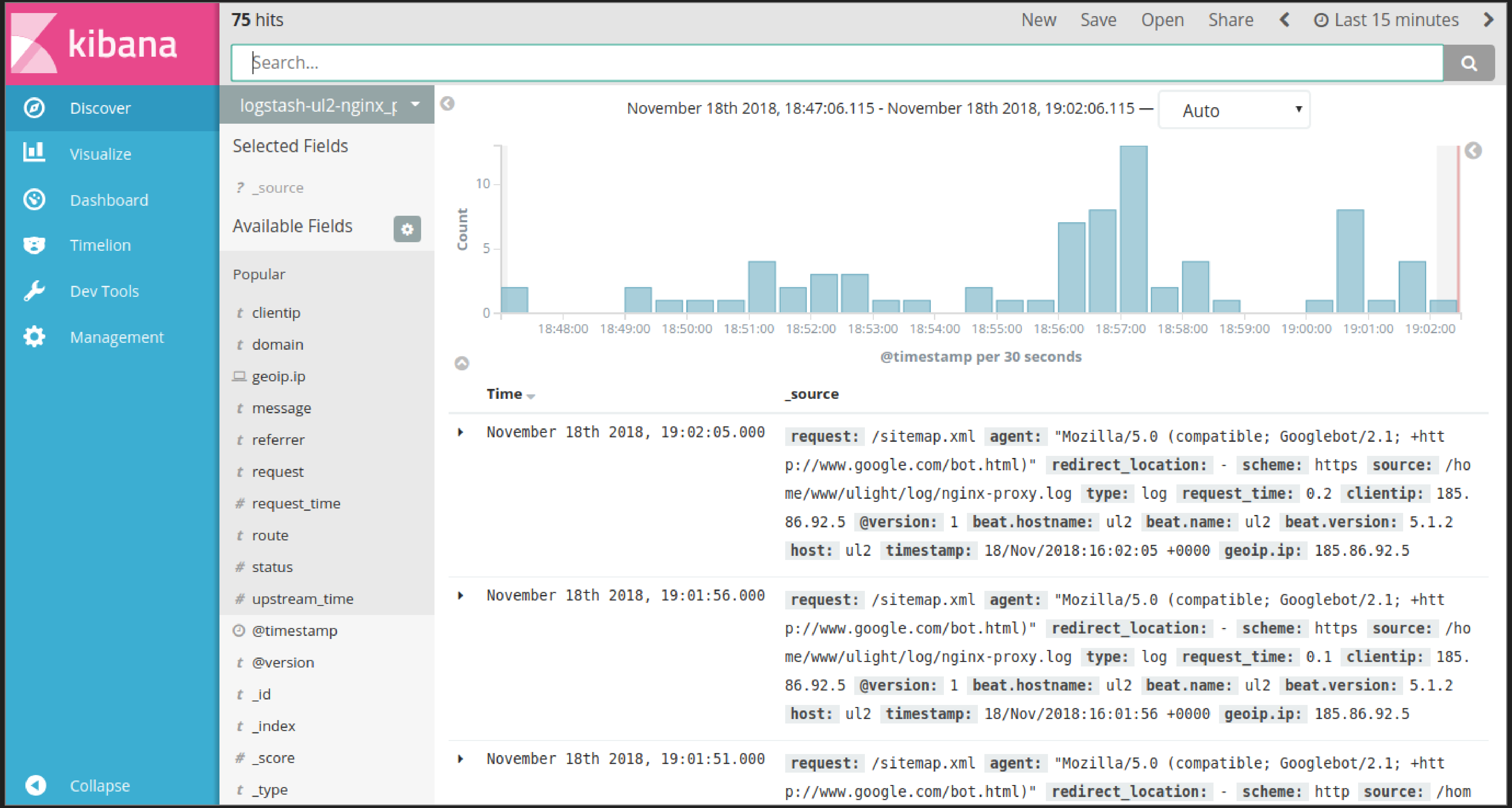
WHEN YOU HAVE STACK... BUT

BOSS



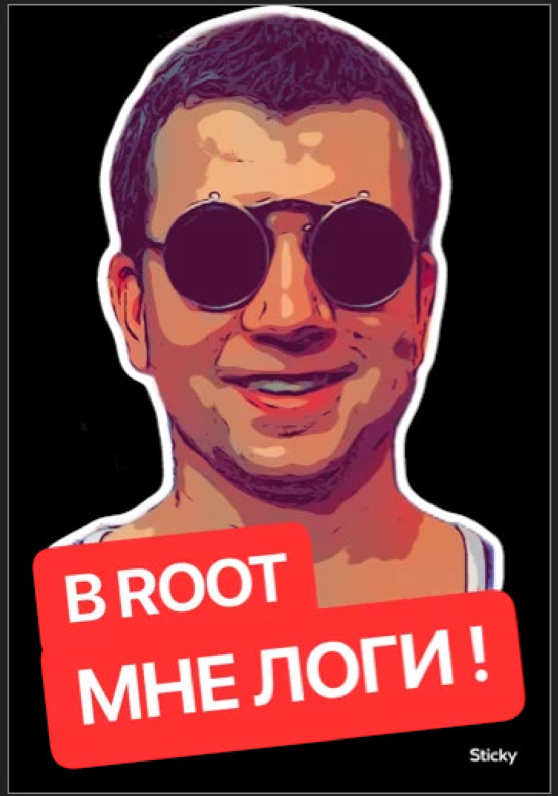
YOU COLLECT ALL AVAILABLE DATA

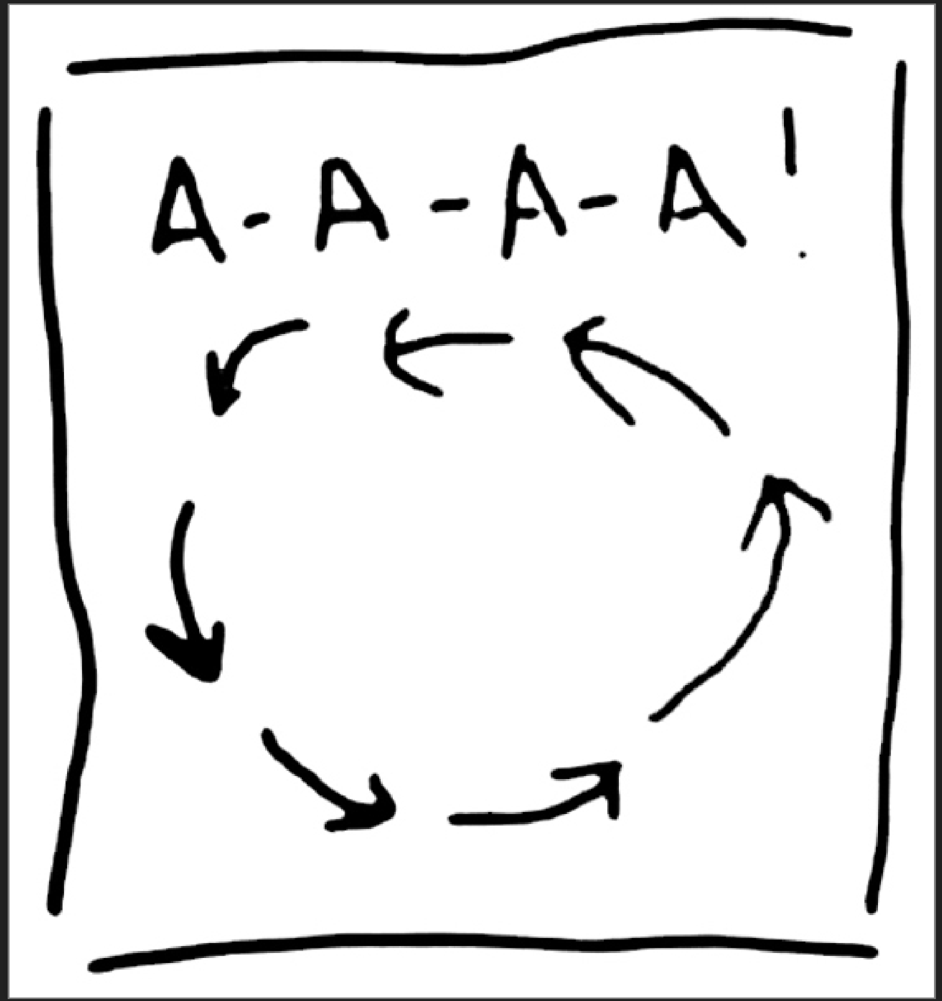


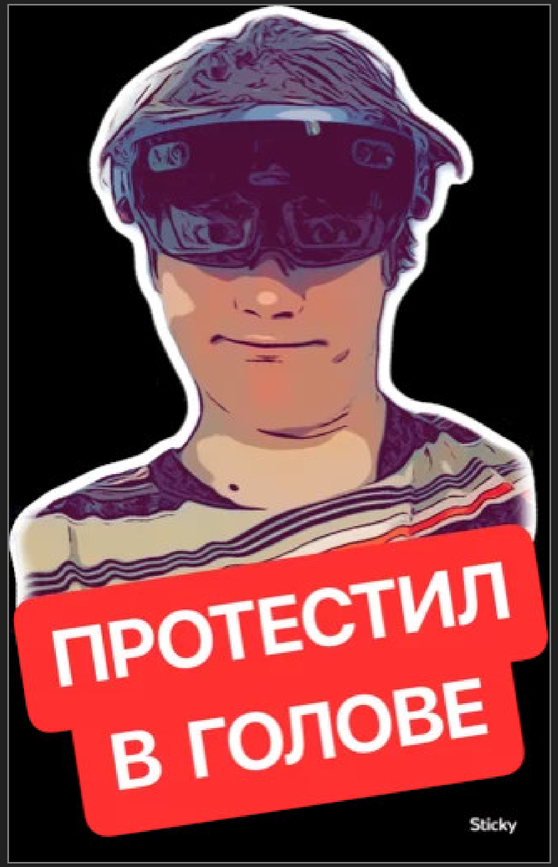


YOU COLLECT ALL AVAILABLE DATA

PREPEARE TEST SAMPLES



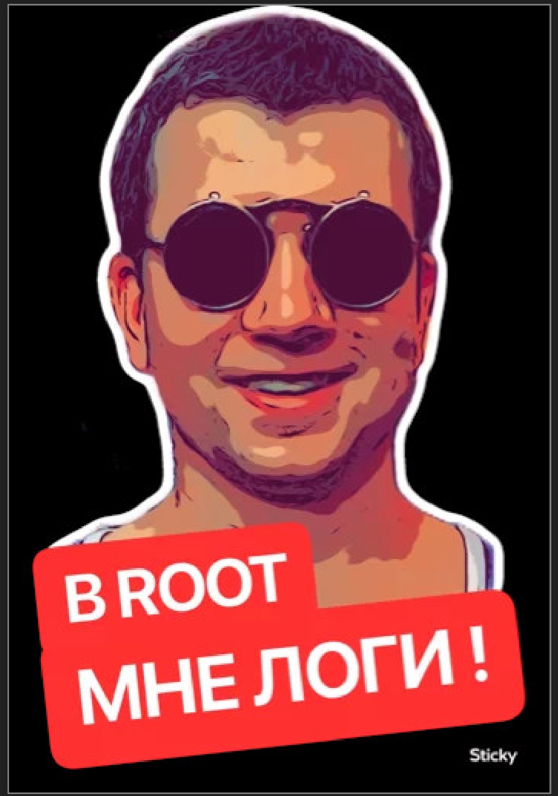


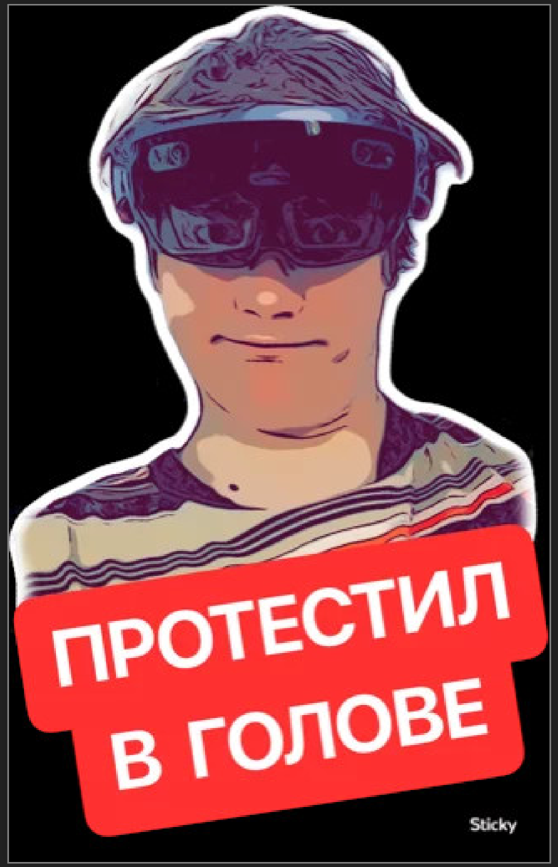


YOU COLLECT ALL AVAILABLE DATA

PREPEARE TEST SAMPLES

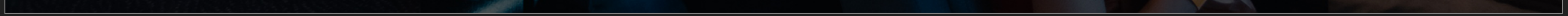
NOTHING HELPS





SO WHAT IS IT ?

SCHRODINGER'S CODE ?



LIVE DEMO !

what the 500 looks like : app.js

now with req.timeout : app.js
enableTimeoout()

WE HAVE TRACE & CODE

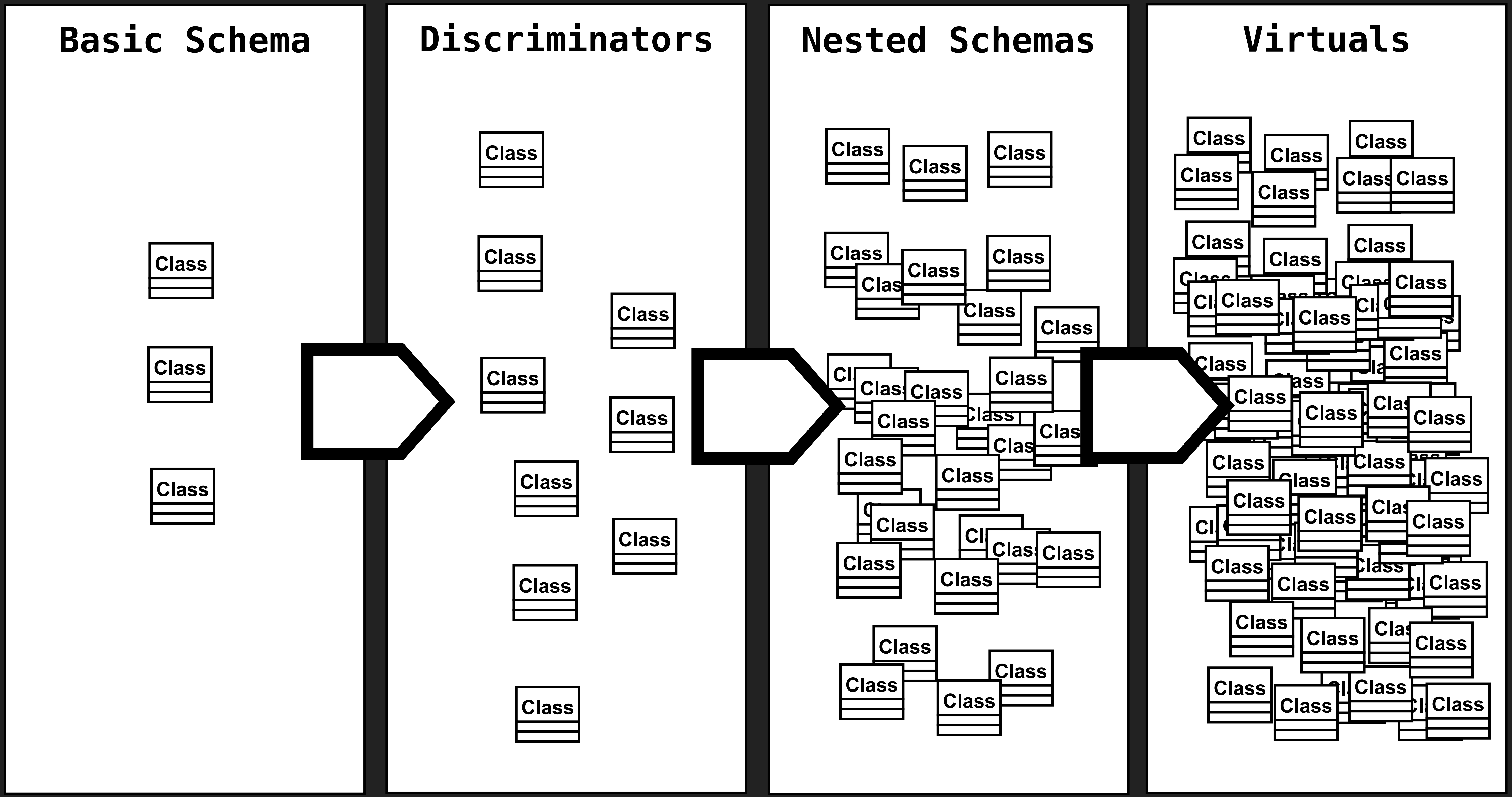
BUT NOT

THE STATE 2 FIX

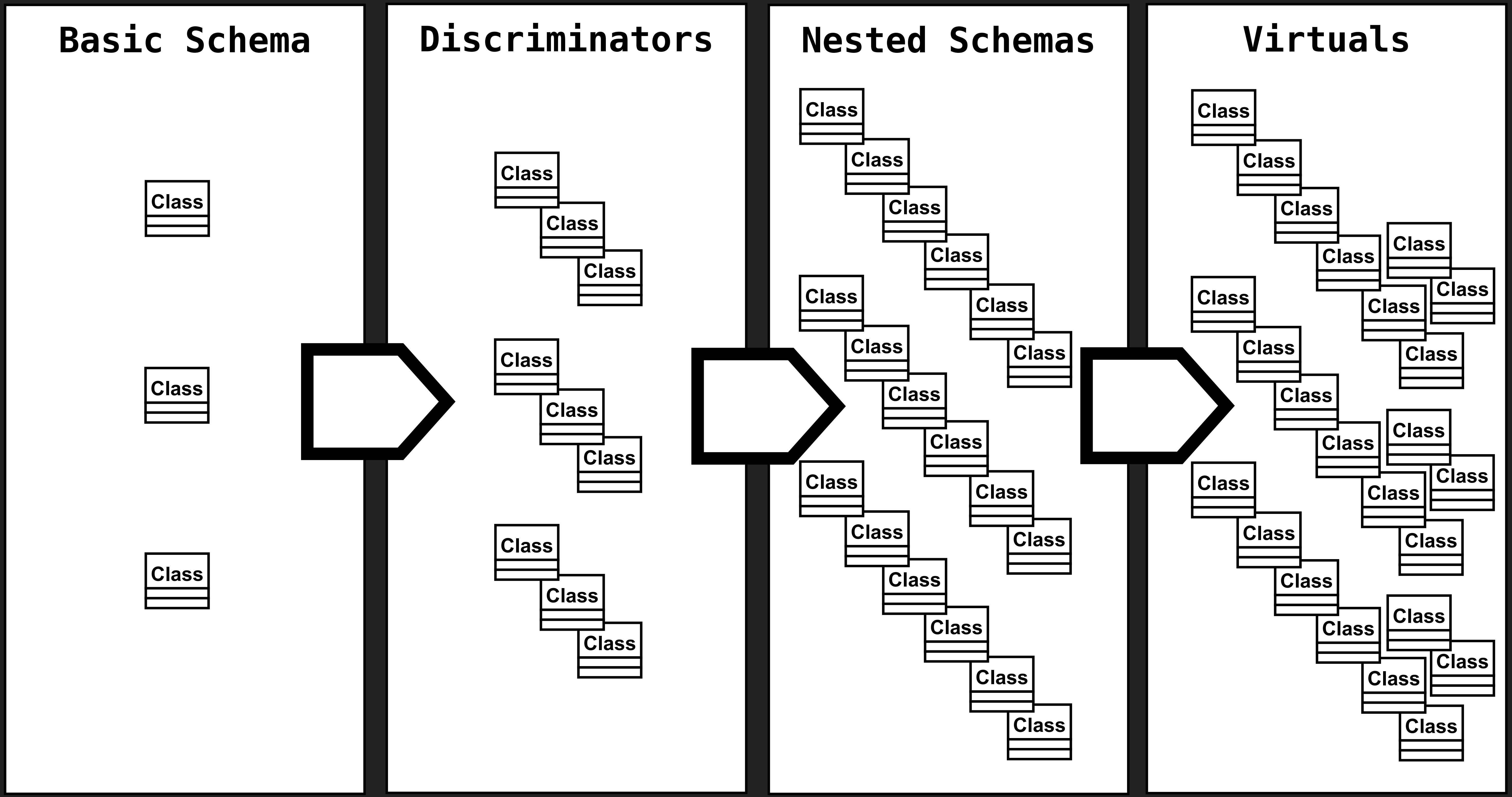
MEANING : WHEN & HOW ?

LET ME DESCRIBE ...

OUR DATA LAYER IS A SORT OF POLYMORPH



IT HAS FRACTAL STRUCT: STI & AOP



BEHAVIOUR DRIVEN BY PROTOTYPE CHAIN

DEPENDS ON SCOPE CONDITIONS

& AIMS COMBINED BVR FOR :

PROTO + STATE + (ARGS)

— SUPERPOSITION MULTIPLIED CONTEXT —

LIVE DEMO !

STATE DRIVEN RESPONSE : FN.BIND(STATE)

STATE DRIVEN 500 STACK : FN.BIND(STATE)

and you can have same code

FOR EXAMPLE

```
> var obj = {};
  obj.foo = 'foo';
  console.log(obj);
  obj.foo = 'bar';

  ▼{foo: "foo"} i
     foo: "bar"
```

AND HAVE STATE DRIVEN ERRORS

```
var count = 0;
const s = () => {
        count++;
        return !!(count%2);
};
// console.log(s());
```

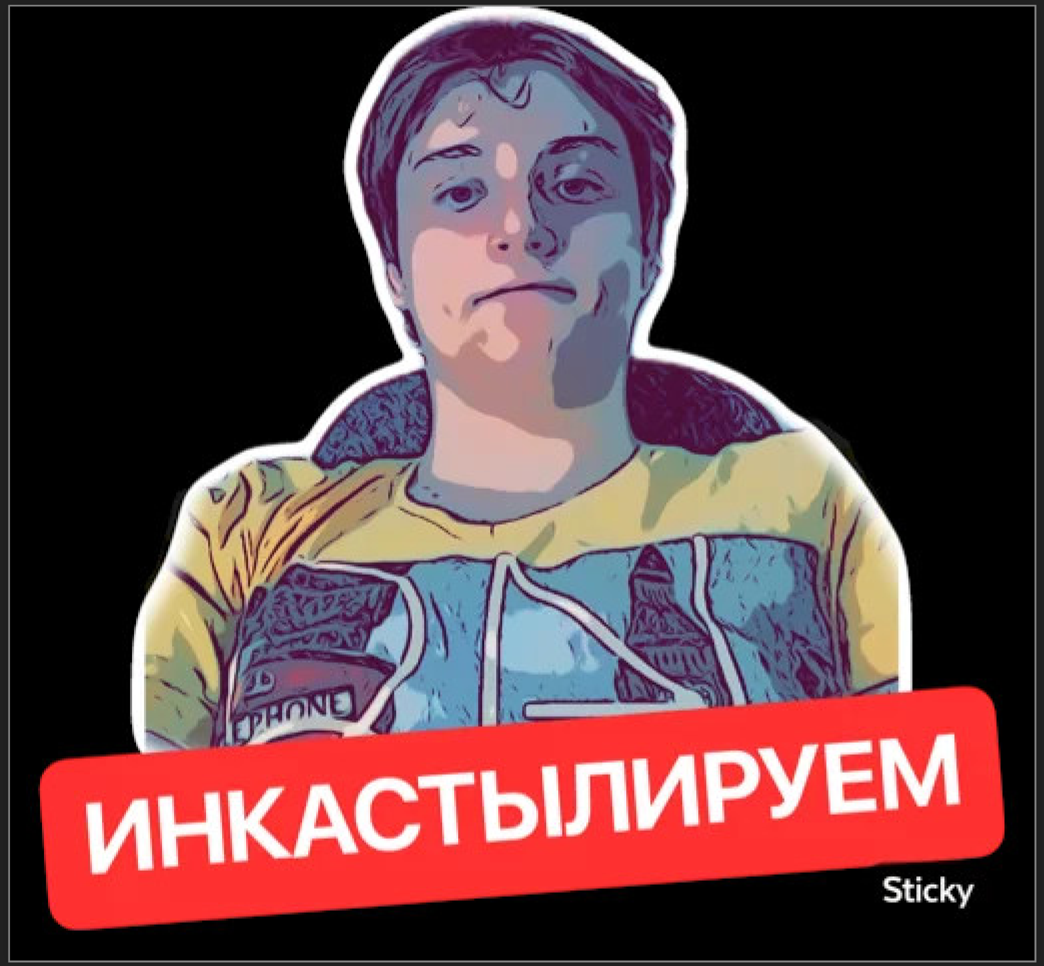
using `console.log`

WE OOPS : CHANGED BVR & STATE

```
var count = 0;
const s = () => {
        count++;
        return !!(count%2);
};
```

```
console.log(s());
if (s()) {
    console.log('Oops!');
}
```

SHOULD MAKE A SPIKE



DIVING

an approach

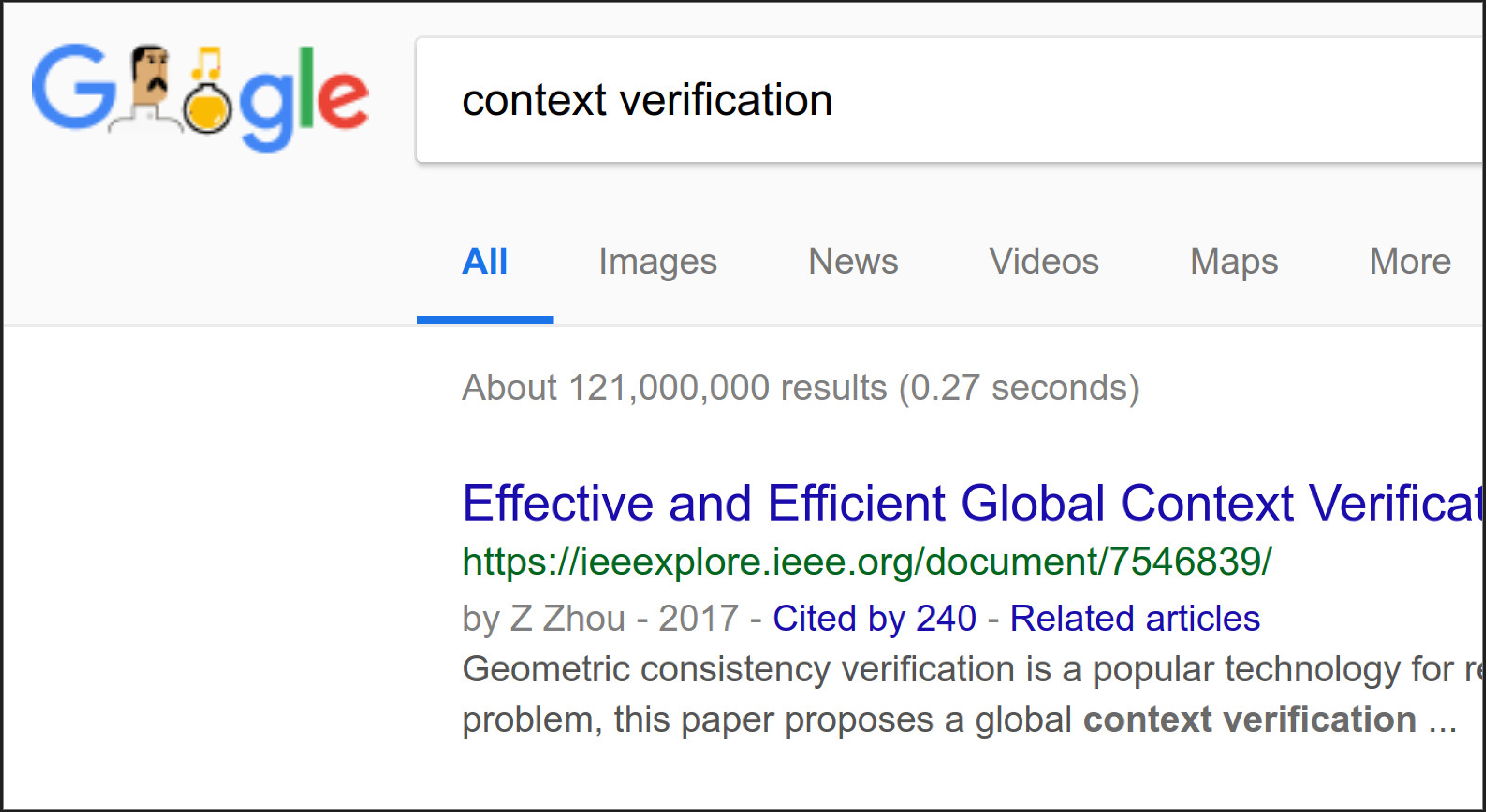
hopless though...

2

NODEJS IS A ROBOT

STATEFULL

CONTEXT MACHINE


context verification
All
Images
News
Videos
Maps
More
About 121,000,000 results (0.27 seconds)
Effective and Efficient Global Context Verificat
https://ieeexplore.ieee.org/document/7546839/
by Z Zhou - 2017 - Cited by 240 - Related articles
Geometric consistency verification is a popular technology for r
problem, this paper proposes a global context verification ...

FORMAL VERIFICATION - WIKI

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help
About Wikipedia
Community portal
Recent changes
Contact page

Article Talk

Formal verification

From Wikipedia, the free encyclopedia

Not to be confused with Verificationism.

"Verifiability" redirects here. For the Wikipedia policy, see Wikipedia:Verifiability. For other uses, see V

This article **needs additional citations for verification**. Please help im
be challenged and removed. *(June 2009) (Learn how and when to remove this*

In the context of hardware and software systems, **formal verification** is the act of proving or disproving th
or property, using formal methods of mathematics.[1]

Formal verification can be helpful in proving the correctness of systems such as: cryptographic protocols,

The verification of these systems is done by providing a formal proof on an abstract mathematical model c
otherwise known by construction. Examples of mathematical objects often used to model systems are: fin
hybrid automata, process algebra, formal semantics of programming languages such as operational sema

CERTIFYING ALGORITHM - WIKI

WIKIPEDIA
The Free Encyclopedia

- Main page
- Contents
- Featured content
- Current events
- Random article
- Donate to Wikipedia
- Wikipedia store

Interaction

- Help
- About Wikipedia
- Community portal

Article Talk

Certifying algorithm

From Wikipedia, the free encyclopedia

In theoretical computer science, a **certifying algorithm** is an algorithm that outputs, together w
efficient if the combined runtime of the algorithm and a proof checker is slower by at most a co

The proof produced by a certifying algorithm should be in some sense simpler than the algorith
algorithm again). Sometimes this is formalized by requiring that a verification of the proof take l
found in linear time) simplicity of the output proof is considered in a less formal sense.[1] For ins
or a checker for the proof may be more amenable to formal verification.[1][2]

Implementations of certifying algorithms that also include a checker for the proof generated by t
run, one of three things happens: it produces a correct output (the desired case), it detects a bu
both the algorithm and the checker are faulty in a way that masks the bug and prevents it from

Graduate Student Solves Quantum Verification Problem

Urmila Mahadev spent eight years in graduate school solving one of the most basic questions in quantum computation: How do you know whether a quantum computer has done anything quantum at all?

... so...

NP-COMPLETENESS

https://en.wikipedia.org/wiki/NP-completeness

SET THEORY

https://en.wikipedia.org/wiki/Set_theory

THEORY OF CONSTRAINTS

https://en.wikipedia.org/wiki/Theory_of_constraints

SELF-CLOCKING SIGNAL

https://en.wikipedia.org/wiki/Self-clocking_signal

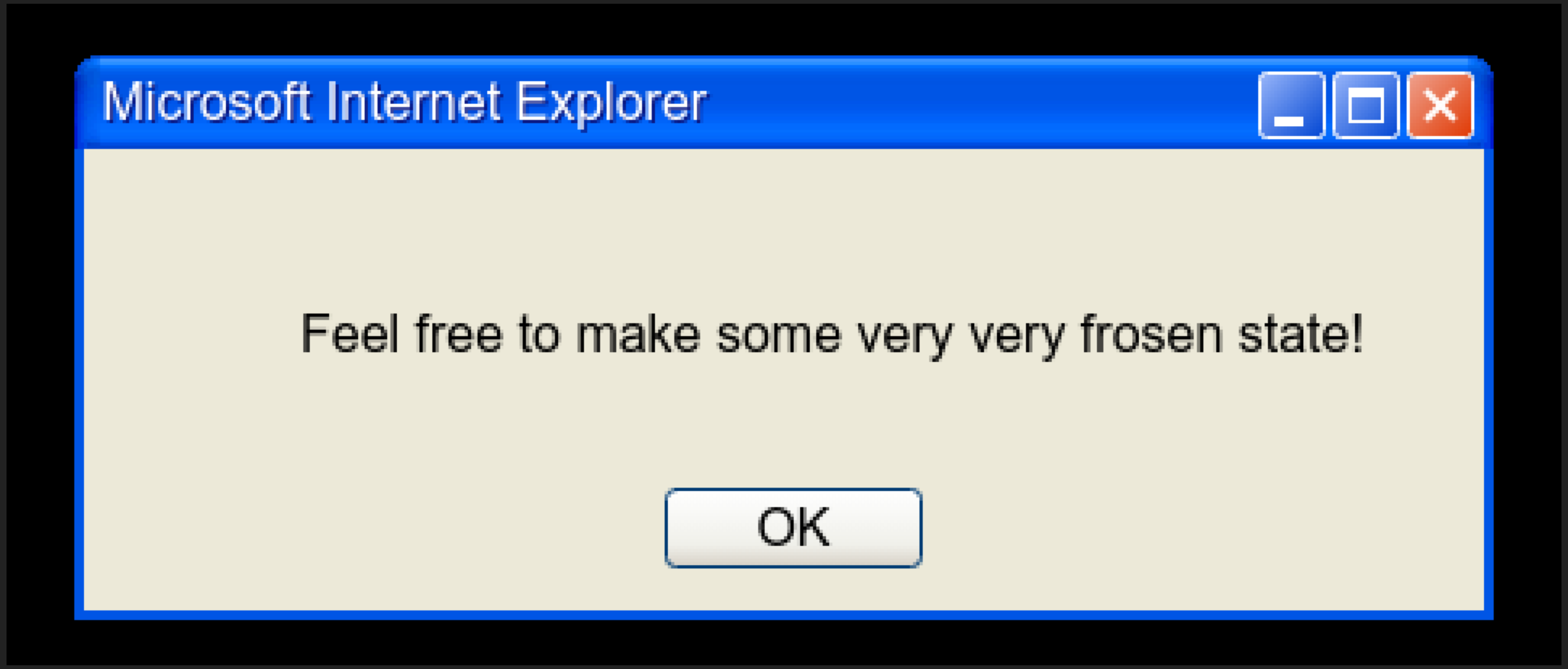
SELF-SYNCHRONIZING CODE

https://en.wikipedia.org/wiki/Self-synchronizing_code

WHAT IT IS ABOUT ?

Networking? Microcontrollers?

HOW SIMPLE IT WAS 20 YEARS AGO



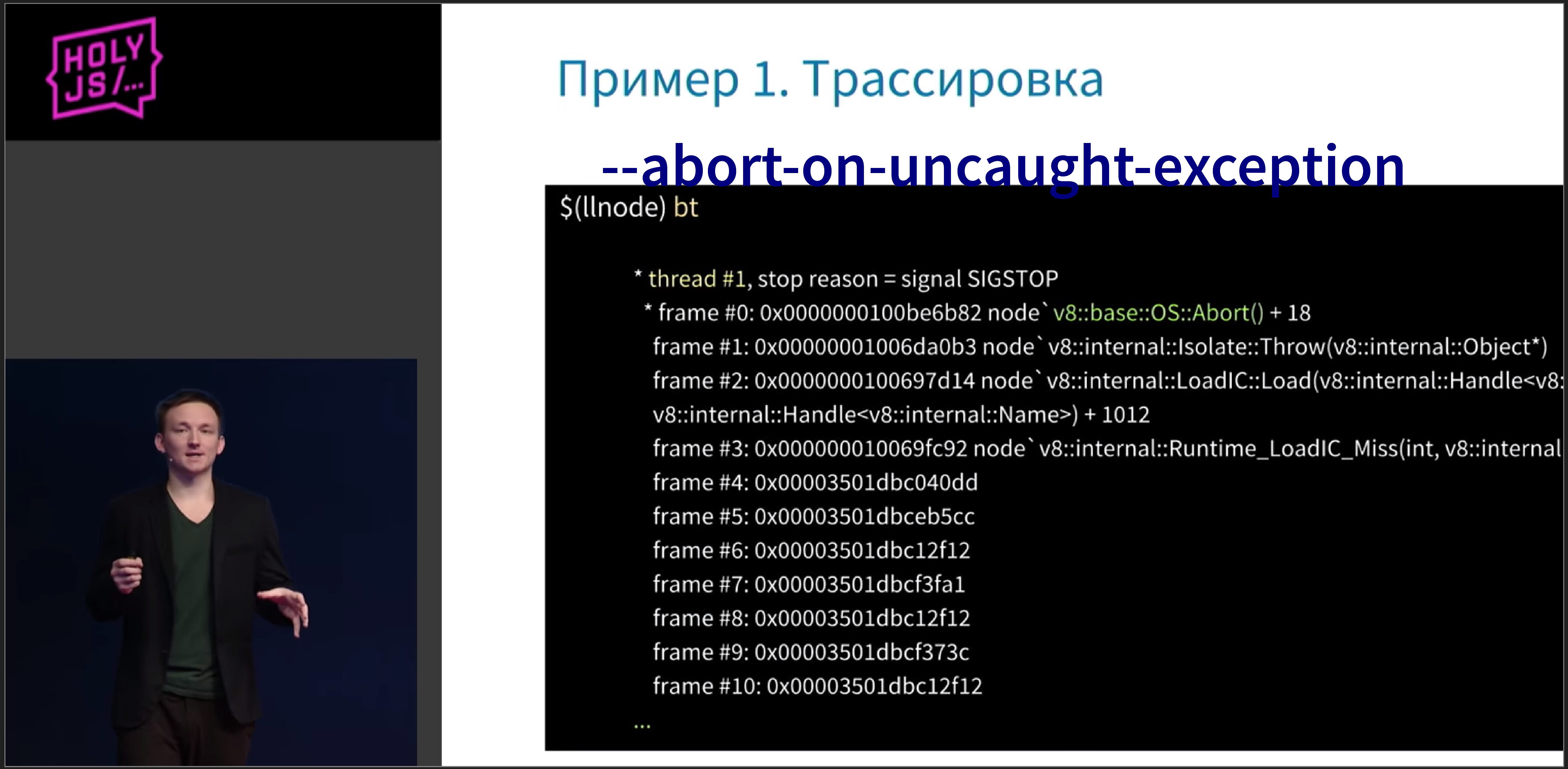
Synchronous request

Note: Starting with Gecko 30.0 (Firefox 30.0 / Thunderbird 30.0 / SeaMonkey 2.27), Blink 39.0, and Edge 13, synchronous requests on the main thread have been deprecated due to the negative effects to the user experience.

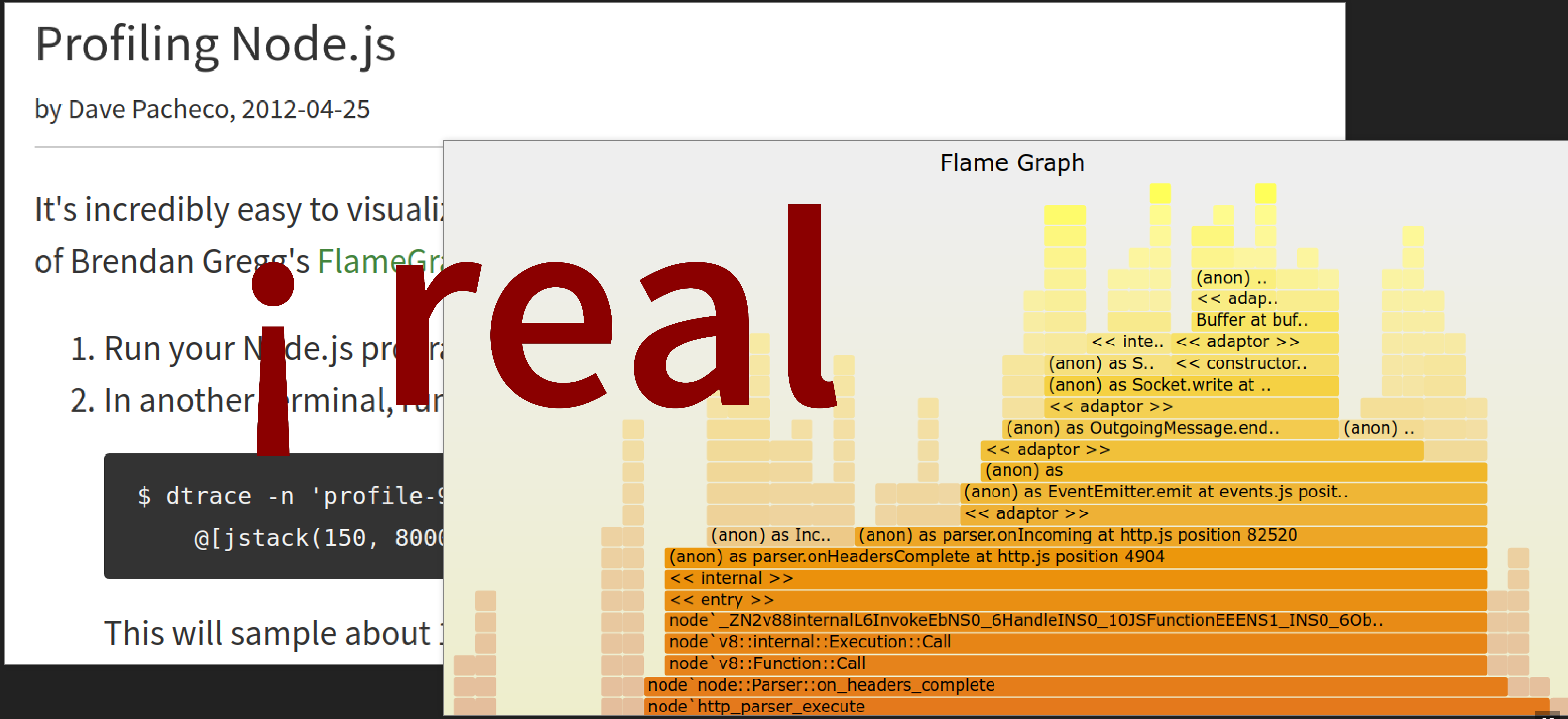
Synchronous XHR often causes hangs on the web. But developers typically don't notice the

WISH I'D ABLE FREEZE THE STATE

CORE DUMP ?

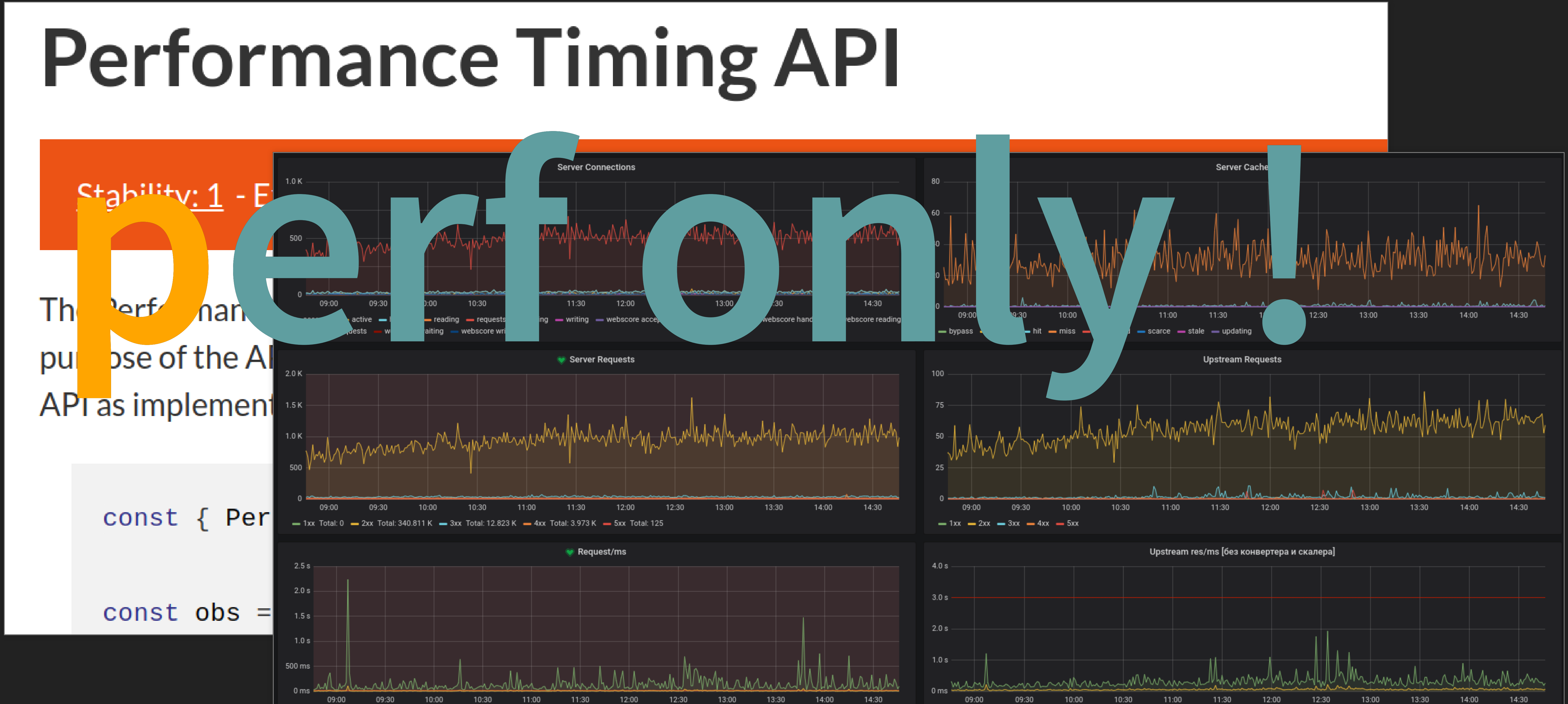

HOLY JS/...
Пример 1. Трассировка
--abort-on-uncaught-exception
$(llnode) bt
* thread #1, stop reason = signal SIGSTOP
* frame #0: 0x0000000100be6b82 node`v8::base::OS::Abort() + 18
frame #1: 0x00000001006da0b3 node`v8::internal::Isolate::Throw(v8::internal::Object*)
frame #2: 0x0000000100697d14 node`v8::internal::LoadIC::Load(v8::internal::Handle<v8:
v8::internal::Handle<v8::internal::Name>) + 1012
frame #3: 0x000000010069fc92 node`v8::internal::Runtime_LoadIC_Miss(int, v8::internal
frame #4: 0x00003501dbc040dd
frame #5: 0x00003501dbceb5cc
frame #6: 0x00003501dbc12f12
frame #7: 0x00003501dbcf3fa1
frame #8: 0x00003501dbc12f12
frame #9: 0x00003501dbcf373c
frame #10: 0x00003501dbc12f12
...

PROFILING & DTRACE ?



WE HAVE GRAPHANA & PROMETSEUS

MAY BE THIS IS IT ?



¡perfonly!

Max Max: 2.237 s Avg: 141 ms Avg Max: 105 ms Avg: 8 ms

max upstream response ms Max: 1.922 s Avg: 317 ms avg upstream response ms Max: 219 ms Avg: 59 ms 3000 Max: 3.000 s Avg: 3.000 s

[[Scope]]

```
▶ constructor: ƒ Function()
  length: 0
  name: ""
▶ toString: ƒ toString()
▶ Symbol(Symbol.hasInstance): ƒ [Symbol.hasInstance]()
▶ get arguments: ƒ ()
▶ set arguments: ƒ ()
▶ get caller: ƒ ()
▶ set caller: ƒ ()
▶   proto  : Object
  [[FunctionLocation]]: <unknown>
▼ [[Scopes]]: Scopes[0]
```

DELETE REQUIRE.CACHE

```
global.reloadModule(filename) {
    delete require.cache[filename];
    return require(filename);
};
```

Node Fibers ...

3.0.0 • Public • Published 3 months ago

Readme | 0 Dependencies

weekly downloads

23,579

version

3.0.0

license

MIT

fibers(1) -- Fiber support for v8 and No

npm v3.0.0 | license MIT | travis passing | downloads 551K/month

Fibers, sometimes called coroutines, are a powerful tool which expose an API to jump between multiple call stacks from within a single thread. This can be useful to make code written for a synchronous library play nicely in an asynchronous environment.

weekly downloads

23,579

version

3.0.0

open issues

9

!prod & expensive

INSTALLING

npm install fibers

0 dependencies   version 3.0.0
233 dependents   updated 3 months ago

THREADS...

Worker Threads

Stability: 1 - Experimental

EXPERIMENTAL

```
const worker = require('worker_threads');
```

--experimental-worker

Workers, unlike child processes or when using the `cluster` module, can also share memory efficiently by transferring `Ar`
instances or sharing `SharedArrayBuffer` instances between them.

CONTROL FLOW

https://en.wikipedia.org/wiki/Control_flow

GREEN THREADS

https://en.wikipedia.org/wiki/Green_threads

RUBY, GO, CLOSURE, LISP ...

SMALLTALK

https://en.wikipedia.org/wiki/Smalltalk

TRACKIG

EVENTS!

EVENTS!

EVENTS!

AN ENTRYPOINT IS ANY OP

FOR EXAMPLE REQUEST

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

app.use('/entrypoint', (req, res) => {
    // do something
});
```

OR FILE READ

```
require('fs').readFile(name, (err, data) => {
    // THIS is ENTRYPOINT TOO
})
```

NO ASYNC 4 ECMASCRIPT® 2015

8.4 JOBS & JOB QUEUES

8.4 Jobs and Job Queues

A Job is an abstract operation that initiates an ECMAScript computation when no other ECMAScript computation is currently in progress. A Job abstract operation may be defined to accept an arbitrary set of job parameters.

Execution of a Job can be initiated only when there is no running execution context and the execution context stack is empty. A PendingJob is a request for the future execution of a Job. A PendingJob is an internal Record whose fields are specified in Table 25. Once execution of a Job is initiated, the Job always executes to completion. No other Job may be initiated until the currently running Job completes. However

is very first note about async tasks
mostly made 4 promises

ALL ASYNCHRONOUS JS

IS A PART OF ENVIRONMENT

- MDN : window.setTimeout
- node : global.setTimeout
- node : process.nextTick

WINDOW

```
window.onerror =
```

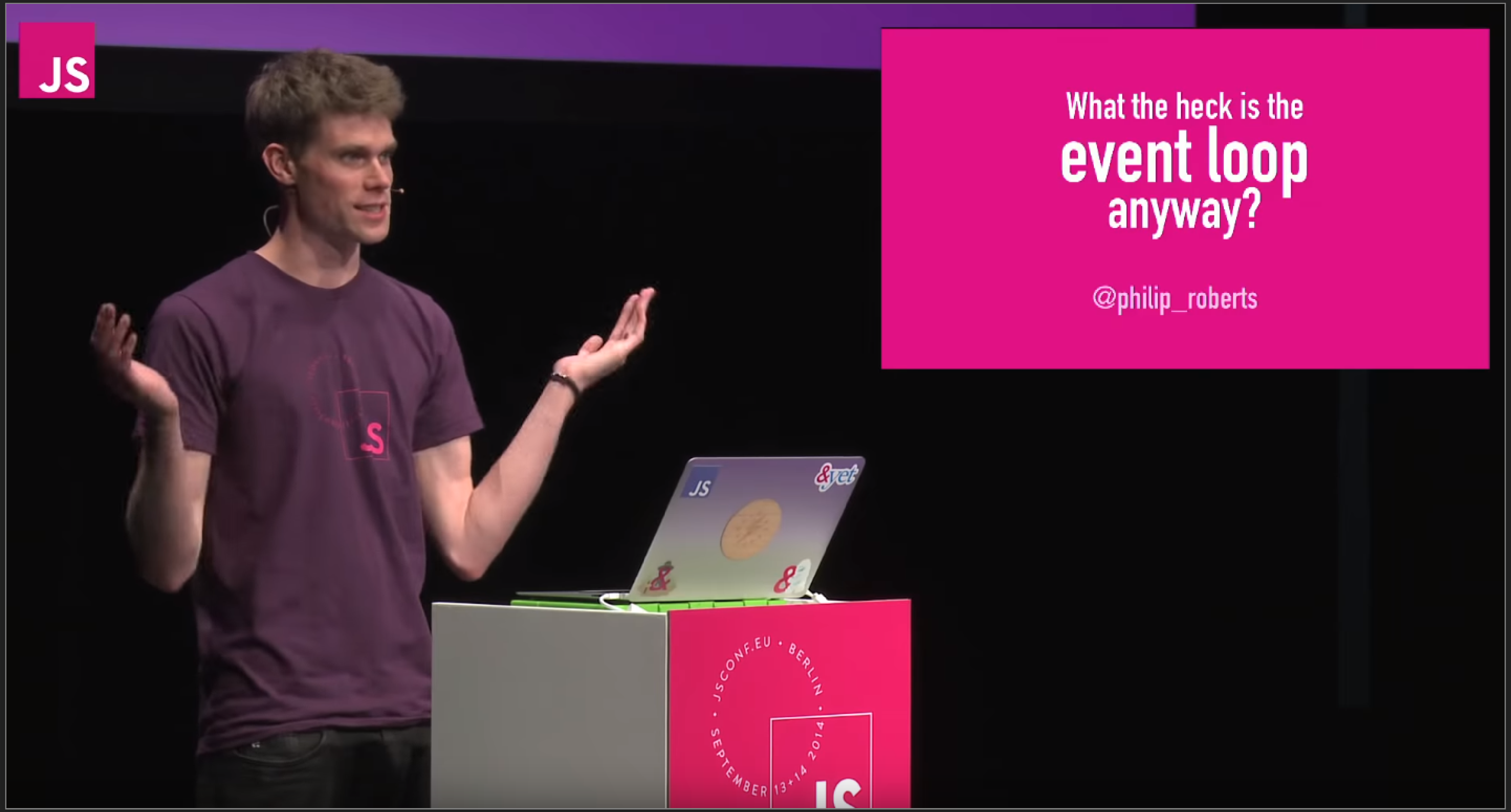
NODE.JS

```
process.on('uncaughtException' ...);

process.on('unhandledRejection' ...);
```

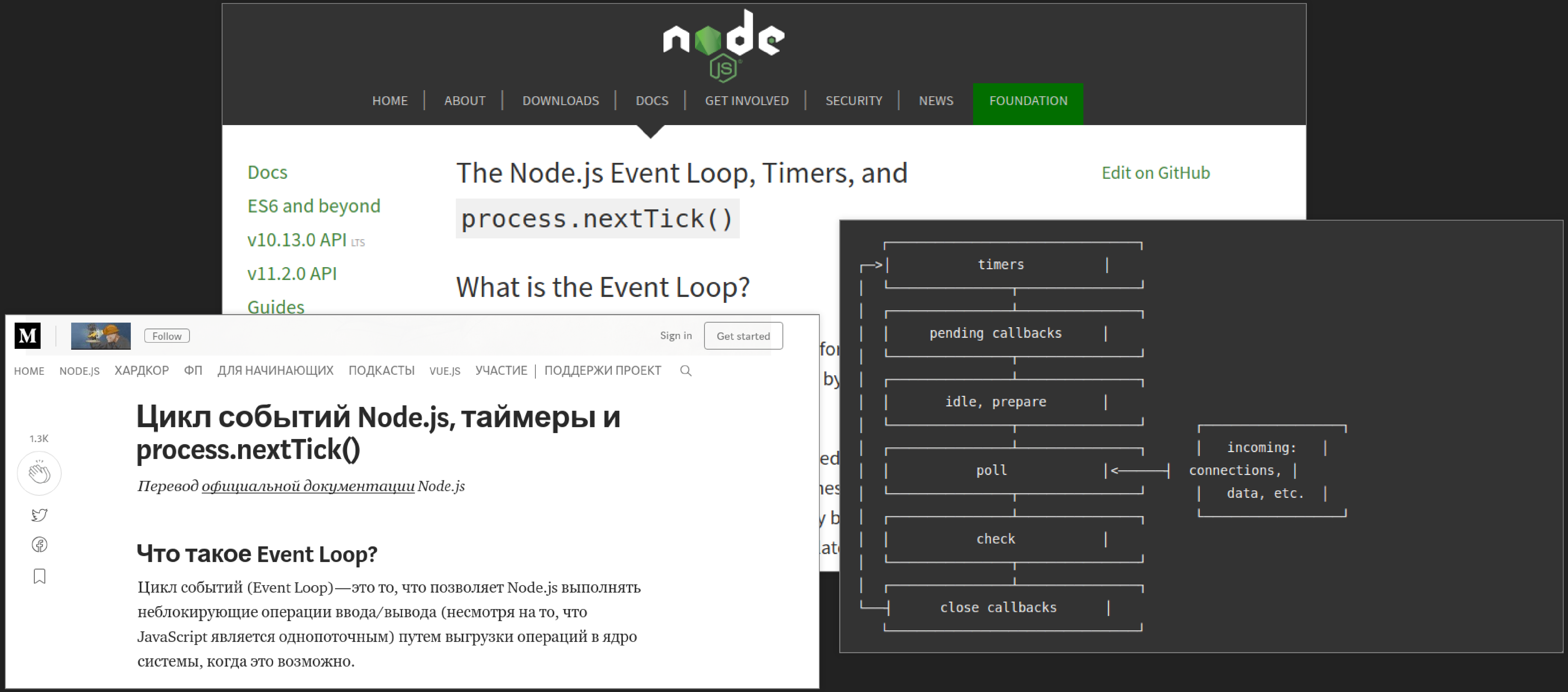
EVENT LOOP

Habr/ YouTube

EVENT LOOP


JS
What the heck is the
event loop
anyway?
@philip_roberts
JSCONF.EU · BERLIN · SEPTEMBER 13+14 2014
JS
&yet

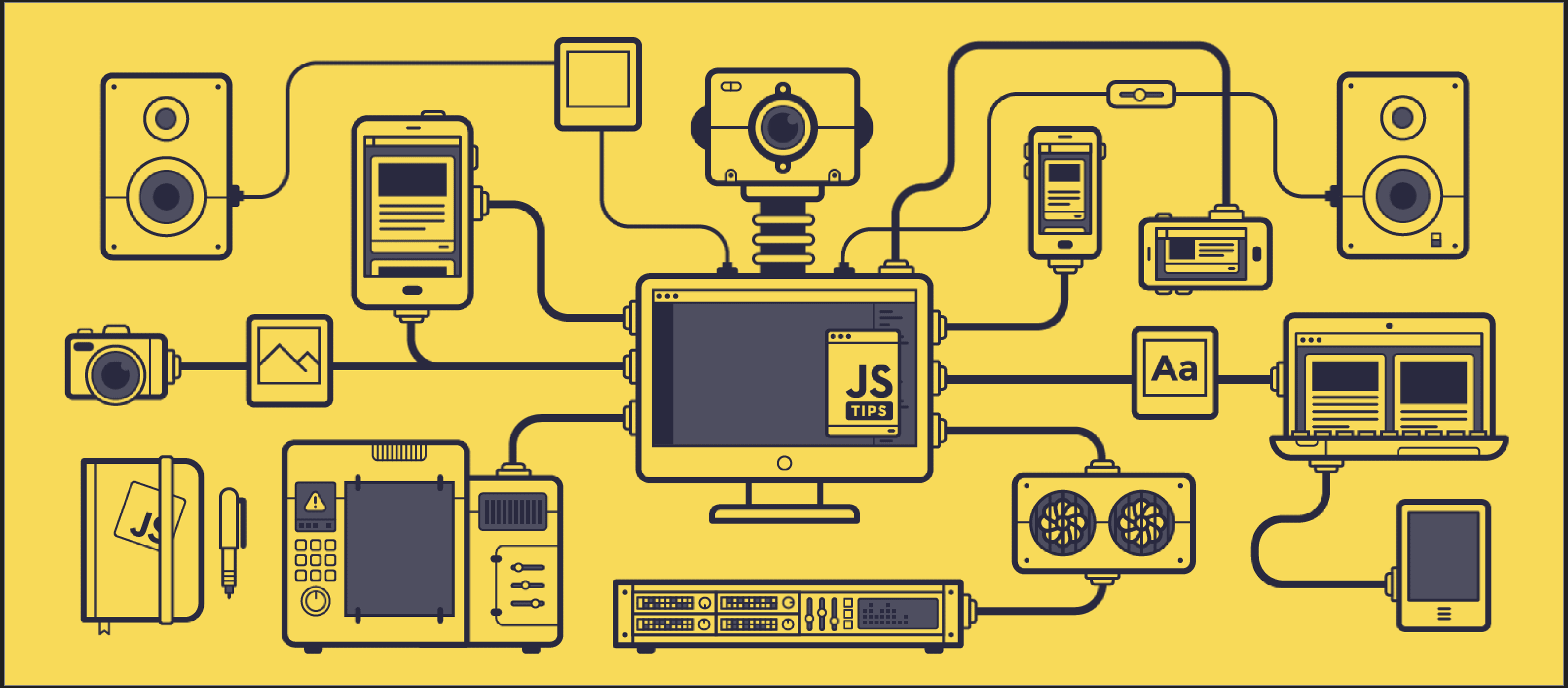
EVENT LOOP : NODE



LIVE DEMO!

setTimeout + fs.readFile: test.js

how we see js — and everything asynchronous



(c) original img article link

BUT INDEED

THERE IS NO ASYNC AT ALL

JUST SEQUENTAL FLOW

VERY FAST ONE, THOUGH

Not logged in Talk Contributions Create account Log in

WIKIPEDIA
The Free Encyclopedia

Article Talk

Read Edit View history

Search Wikipedia

Von Neumann architecture

From Wikipedia, the free encyclopedia

and Univers
nown as the
er architectu
ysicist John
AC.[1] That d
al computer
thmetic logi
tion register
ons

EVERY ASYNC OPERATION

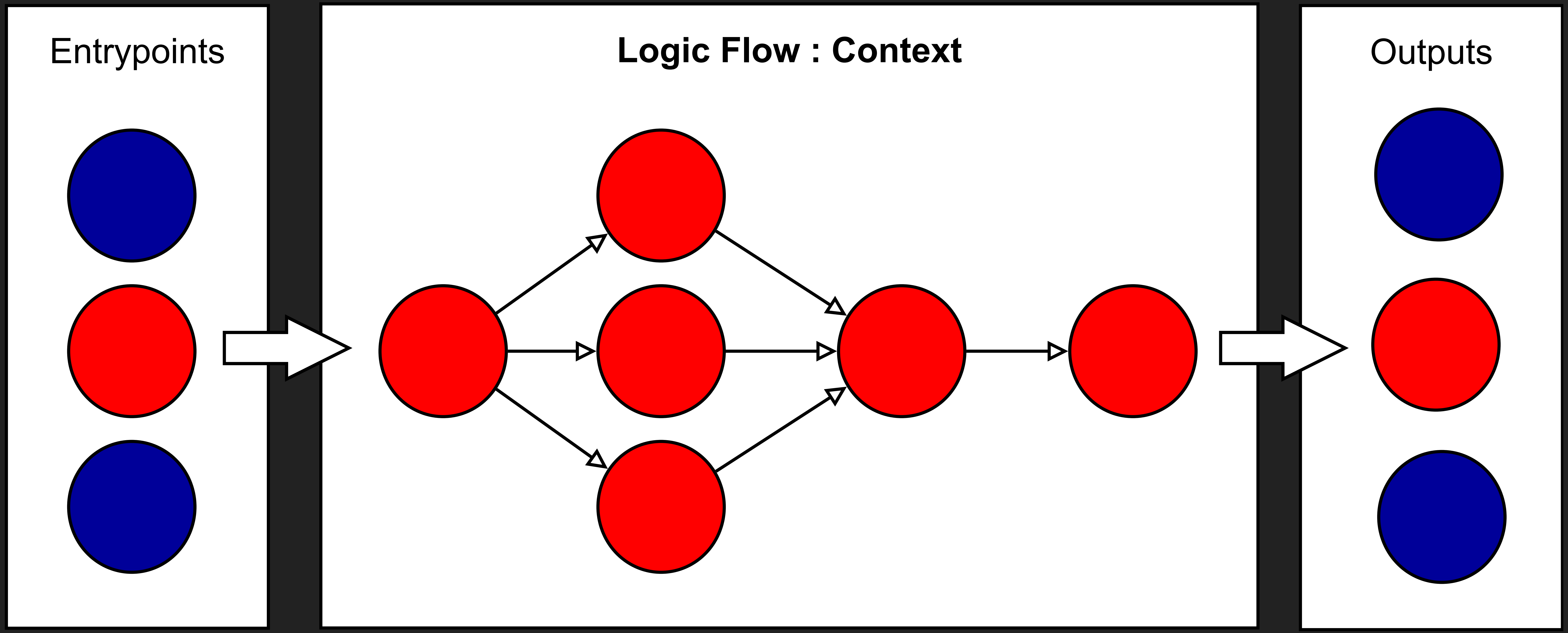
JUST TRIGGERS

INTERRUPTION CALLBACK

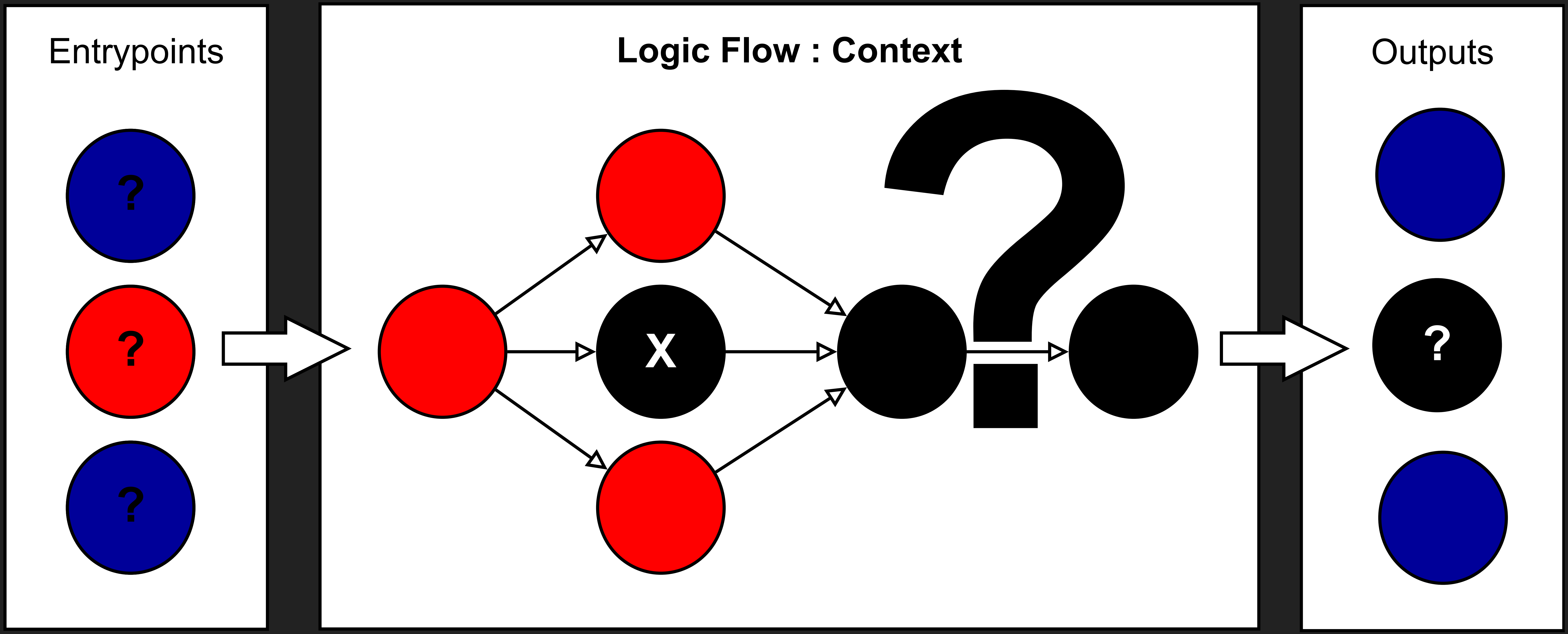
REGISTERED SOMEWHERE

AS A CONTINUATION CODE

LOGICAL CONTEXT EXPLANATION



LOGICAL CONTEXT FAILURE



AND WE USUALLY USE

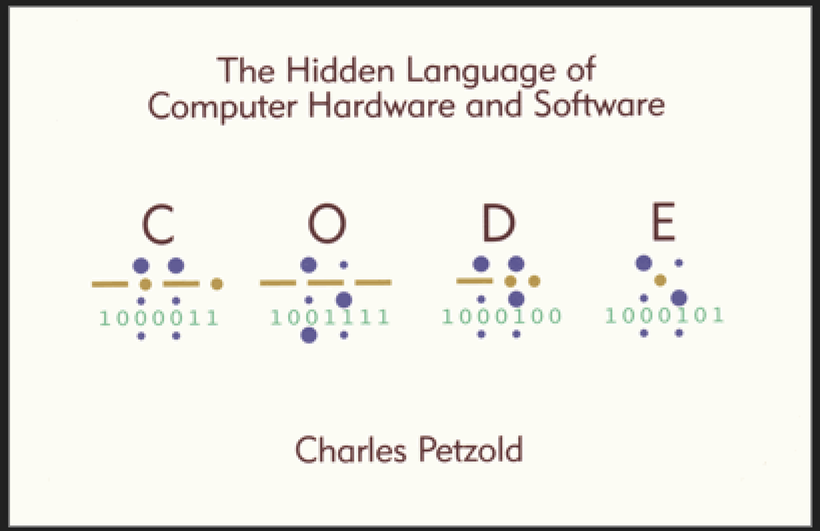
something like this 2 contextify the continuation

```
fn = async (req, ...args) => { /* ... */ };

        /* ... ¯\_(ツ)_/¯ ... */

fn.bind(context, ...args) => { /* ... */ };
```

& no way 4 .uncaughtException

```
process.on('uncaughtException', (err) => {
    log.error(err);

    // ??? req.res.status(503);

    process.exit(1);
});
```

C.0.D.E.



The Hidden Language of
Computer Hardware and Software
C O D E
1000011
1001111
1000100
1000101
Charles Petzold

IMAGINE

CODE WITH NO LIMITS

Async Hooks

[Stability: 1](#) - Experimental

The `async_hooks` module provides an API to register callbacks tracking the lifetime of asynchronous resources created inside a Node.js application. It can be accessed using:

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

no more flag : lets try em

.init (asyncId, type,
triggerId, resource)

.before (asyncId)

.after (asyncId)

.delete (asyncId)

.promiseResolve (asyncId)

asyncId

triggerId

async_hooks.executionAsyncId()

async_hooks.triggerAsyncId()

type & resource

DOCUMENTATION EXAPMLE

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

```
const hooks = async_hooks.createHook({
    init(asyncId, type, triggerAsyncId, resource) {
        /* some code */
    }

    before(asyncId) { /* some code */ }

    after(asyncId) { /* some code */ }

    destroy(asyncId) { /* some code */ }
});
```

```
hooks.enable();
```

DOCUMENTATION EXAPMLE

```
FSEVENTWRAP, FSREQCALLBACK, GETADDRINFOREQWRAP, GETNAMEINFOREQWRAP, HTTPPARSER,
JSSTREAM, PIPECONNECTWRAP, PIPEWRAP, PROCESSWRAP, QUERYWRAP, SHUTDOWNWRAP,
SIGNALWRAP, STATWATCHER, TCPCONNECTWRAP, TCPSERVERWRAP, TCPWRAP, TTYWRAP,
UDPSENDWRAP, UDPWRAP, WRITEWRAP, ZLIB, SSLCONNECTION, PBKDF2REQUEST,
RANDOMBYTESREQUEST, TLSWRAP, Microtask, Timeout, Immediate, TickObject
```

```
TCPSERVERWRAP(5): trigger: 1 execution: 1
TickObject(6): trigger: 5 execution: 1
before:  6
  Timeout(7): trigger: 6 execution: 6
after:   6
destroy: 6
before:  7
>>> 7
  TickObject(8): trigger: 7 execution: 7
after:   7
before:  8
after:   8
```

SO THIS EXAMPLE SHOULD WORK

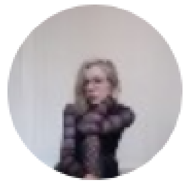
```
const fn = (cb) => {
    setTimeout(() => {
        nextTick(() => {
            Promise
                .then(() => {
                    // tracked !!!
                    cb();
                })

        })
    });
};

const cb = () => { /* some code */ };
```

... SEEMS VERY NICE ...

LET READ SOME ARTICLES

SRSLY, WHO AND WHERE CARES OF BORING DOCS

MY ERROR # 1



Irina Shestak Follow
javascript, hot takes, 80% [].
Aug 28, 2017 · 3 min read

async_hooks in node.js, illustrated

async_hooks, the new and still experimental API that came out with node 8 got my attention a few weeks back, so I decided to do a bit of exploring as to what it can help me do.

async_hooks API essentially makes it easier for you to track your resources (bye `dtrace` ?). You start off by initializing it with an object of callbacks: `init` , `before` , `after` and `destroy` .

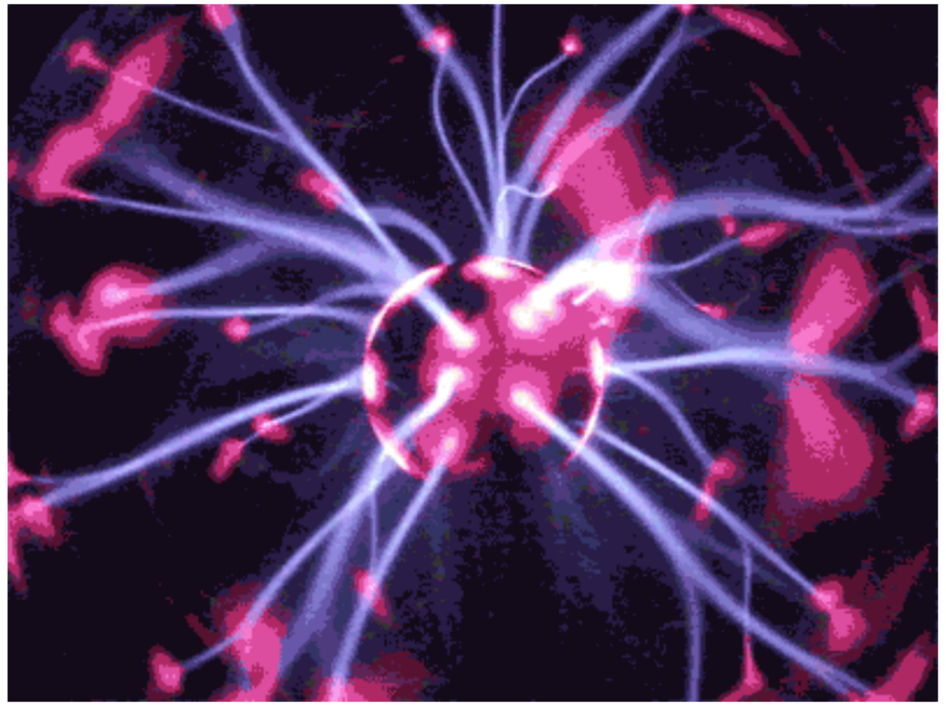
NODECONF EU

Thorsten Lorenz, NodeSource

Grokking Asynchronous Work in Node.js

@thlorenz

Why Async Hooks



- some decided ...
- **We can do better** ...

PROBLEMS...

WHY THIS TALK TITLE ?

context.js

```
0

const eid = hooks.eid;   eid = 1
const tid = hooks.tid;   tid = 0


debugger;
```

SEARCH AGAIN ...

— EXECUTION CONTEXT IS : CLS —

Continuation or Thread Local Storage (CLS)

[C || T]LS - wikipedia quote summary for JS :

method that binds some data
to a logical execution thread

WHAT CLS GIVES

- meaningfull uncaughtException notes
- long stack traces longjohn
- async events marks
- tracing pub~sub
- simultaneous
  shared memory
  calculations

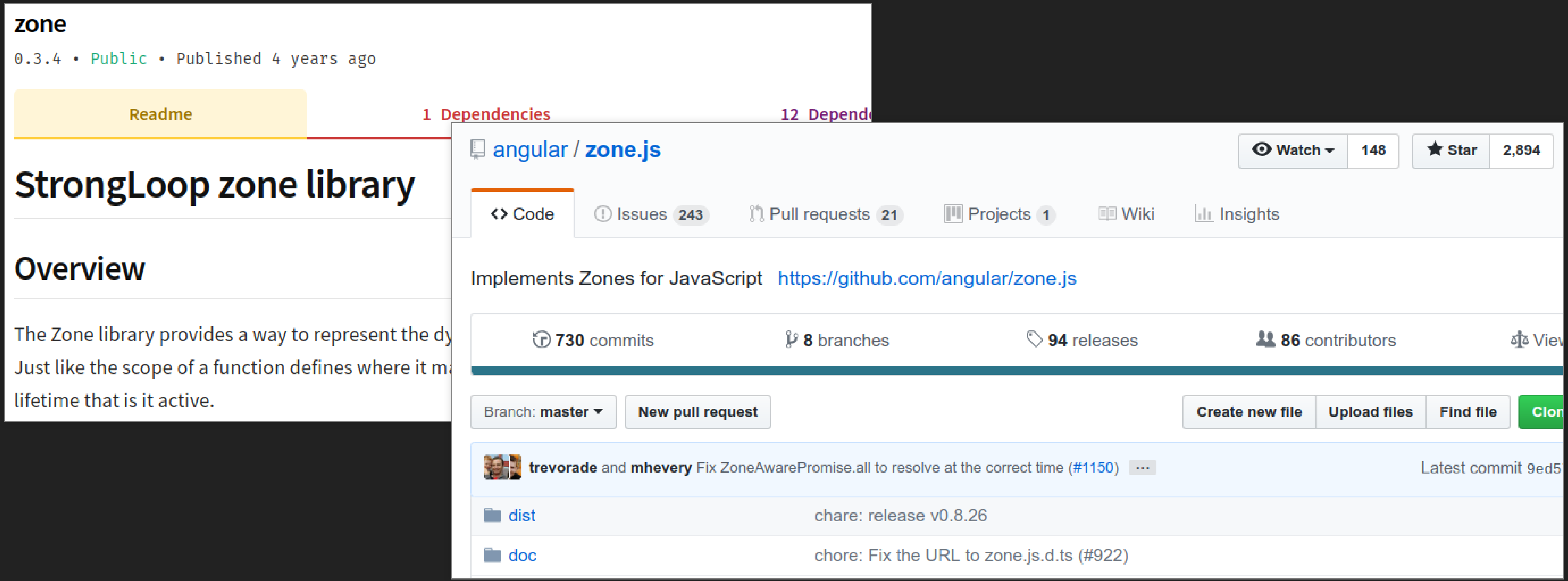
— CLS : FROM NPM —

Let search for this idea at npm : I'm not the only one !

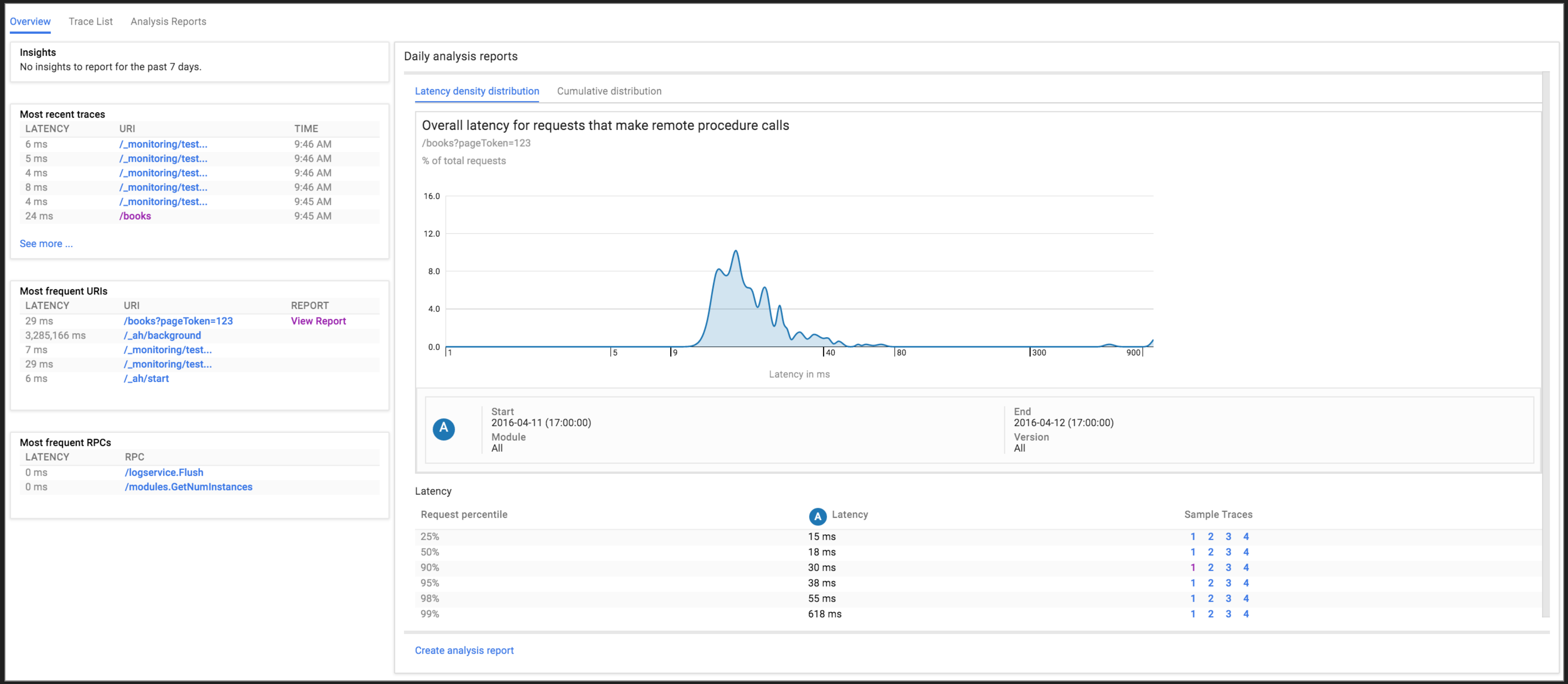
```
$ npm search cls | wc -l
21

$ npm search scope  | wc -l
31

$ npm search context  | wc -l
26

$ npm search continuation  | wc -l
25
```

BROWSER TOOLS : ZONE.JS

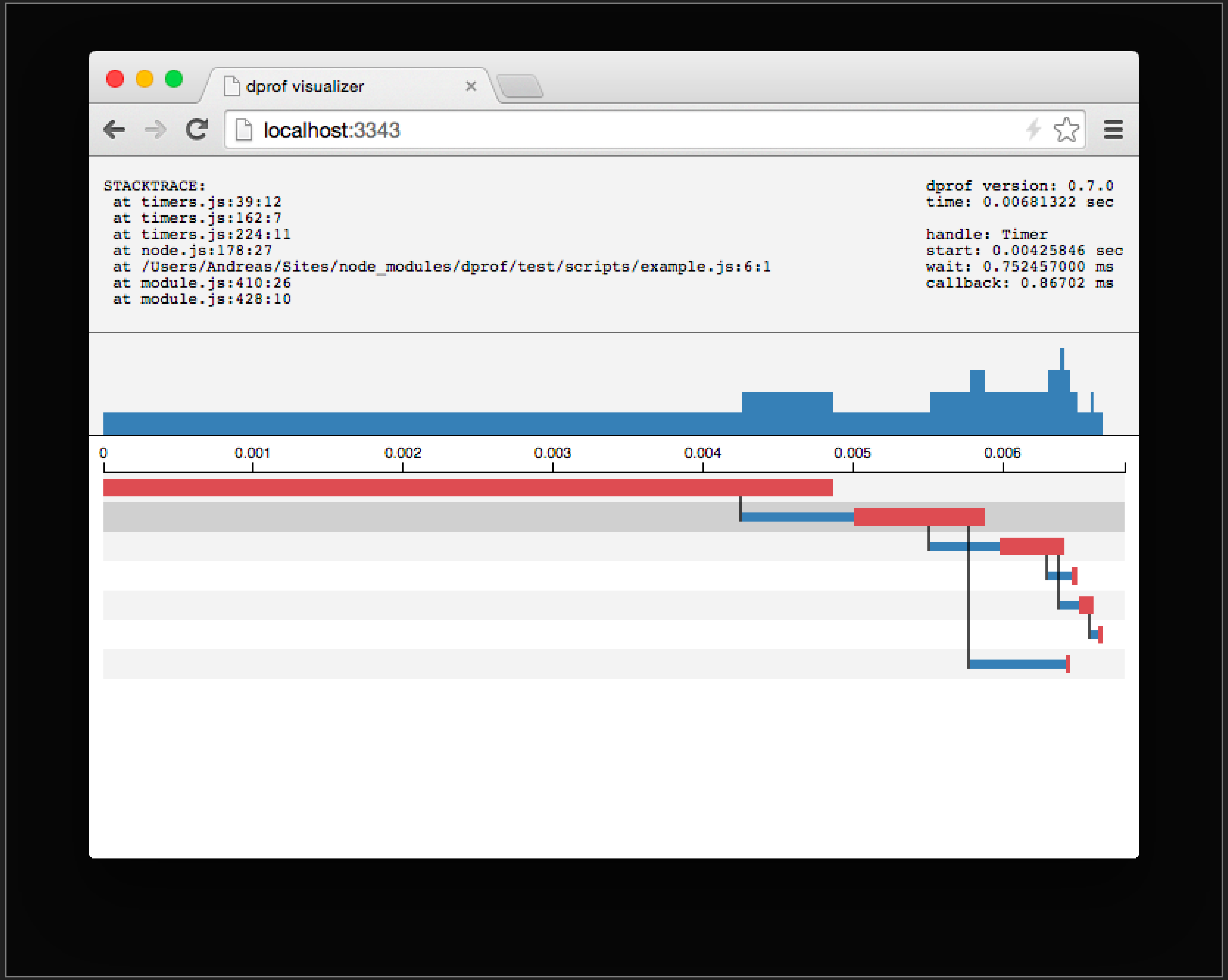
REMOVED FROM ANGULAR



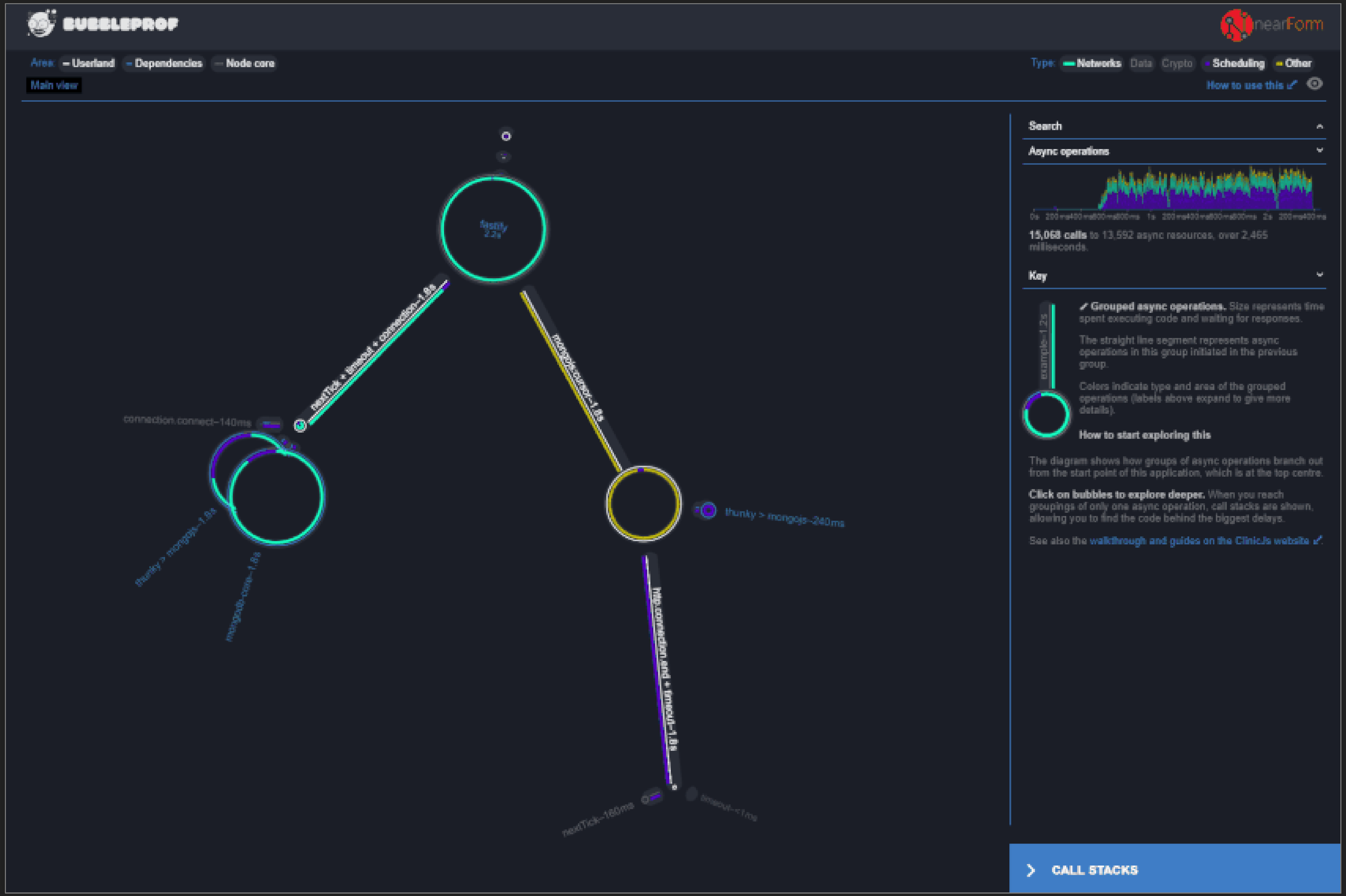
GOOGLE CLOUD TRACING CLS

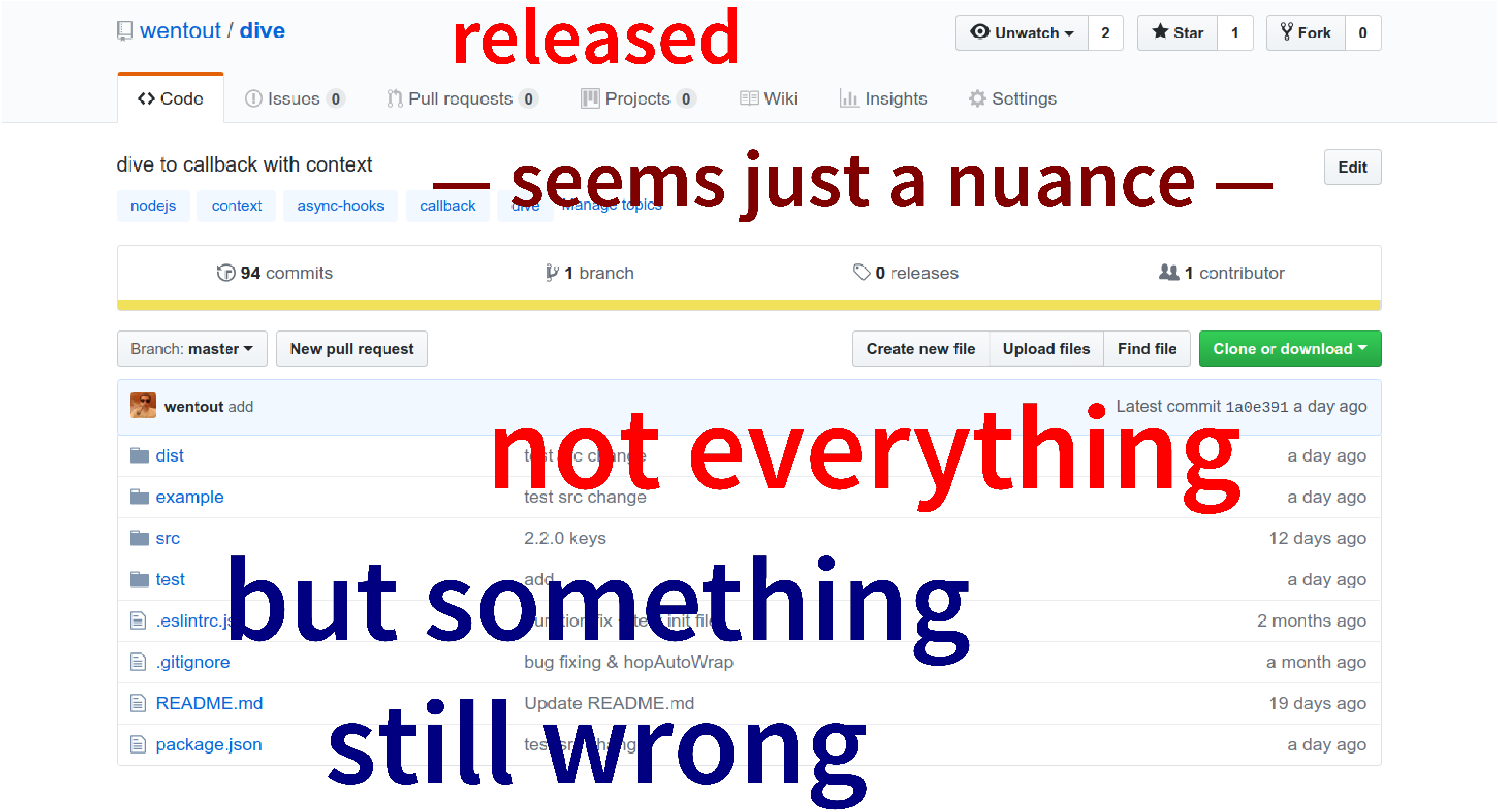


ANDREAS MADSEN DPROF



CLINC + BUBBLEPROF

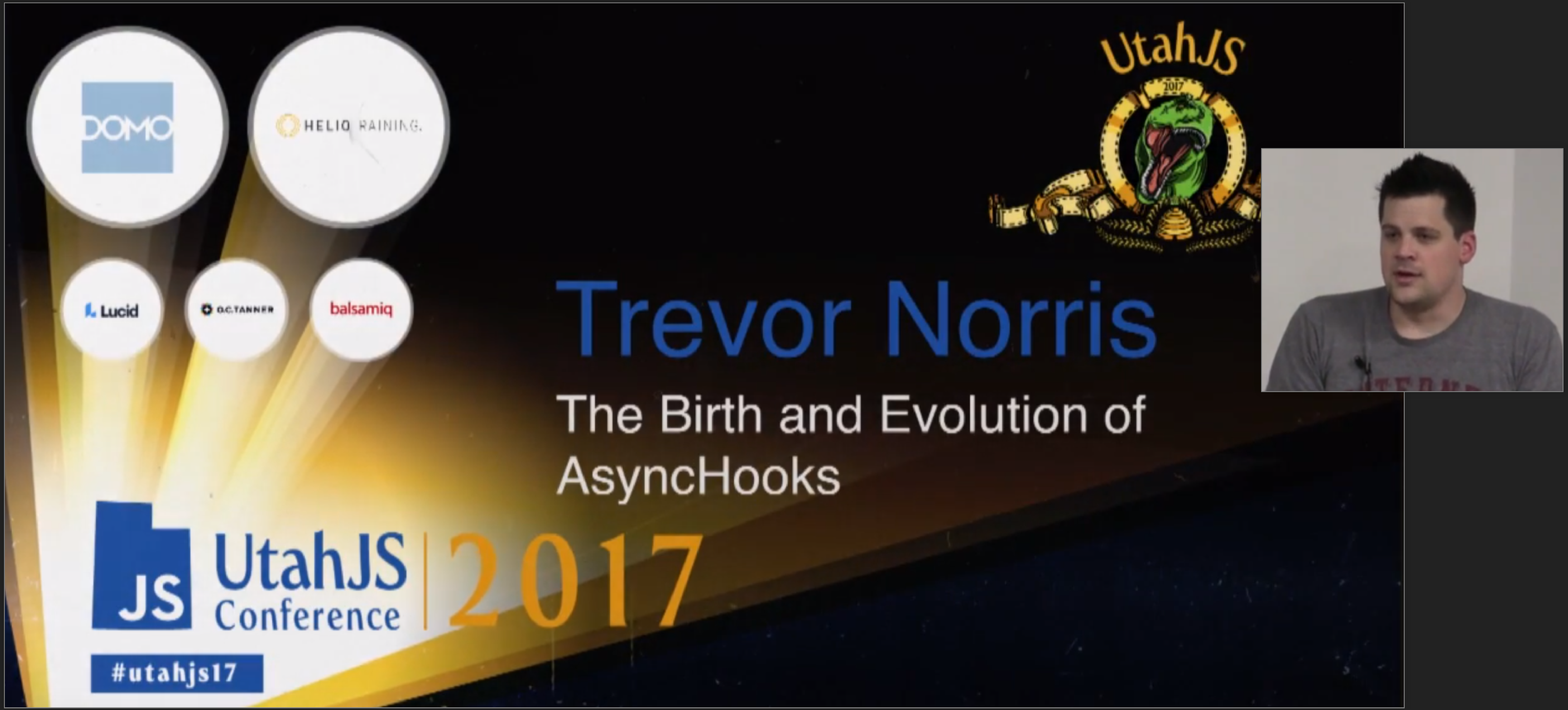



released
— seems just a nuance —
not everything
but something
still wrong
wentout / dive
Unwatch 2
Star 1
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
dive to callback with context
Edit
nodejs
context
async-hooks
callback
dive
Manage topics
94 commits
1 branch
0 releases
1 contributor
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
wentout add
Latest commit 1a0e391 a day ago
dist
a day ago
example
test src change
a day ago
src
2.2.0 keys
12 days ago
test
add
a day ago
2 months ago
.gitignore
bug fixing & hopAutoWrap
a month ago
README.md
Update README.md
19 days ago
package.json
a day ago

LIMITATIONS

5

HOW IT STARTED : ASYNC-LISTENER



FURTHER DEVELOPMENT

Andreas Madsen
AndreasMadsen

Unfollow

Block or report user

nearForm
Copenhagen, Denmark
amwebdk+github@gmail.com

Overview | Repositories 129 | Stars 202 | Followers 211 | Following 45

Popular repositories

trace
Creates super long stack traces
JavaScript ★ 159 ⑂ 13

clarify
Remove nodecore related stack trace noise
JavaScript ★ 110 ⑂ 19

dprof
Dynamic/structured profiling & visualization for sync and async operations
JavaScript ★ 64 ⑂ 11

htmlparser-benchmark
Simple benchmark for diffrent htmlparsers using real-life data
HTML ★ 36 ⑂ 3

stack-chain
API for combining call site modifiers
JavaScript ★ 34 ⑂ 7

async-hook
Inspect the life of handle objects in node
JavaScript ★ 32 ⑂ 13

HOW IT'S MADE

```
var counter = 0;
var currentId = 0;

    var hookSetTimeoout = function (cb, time, ...params) {
    counter++;

    hooks.init.forEach(function (hook) {
        hook(counter, currentId, cb);
    });

    var hooksWrapper = function (currentId, hook) {
        hooks.before.forEach( /*... */ );

        cb.call(null, ...params); // !!!

        hooks.after.forEach( /*... */ );
    }.bind(null, counter);

    setTimeout(hooksWrapper, time);
};
```

WHAT WE CAN & WHAT NOT

```
process.binding('async_wrap') // LIVE DEMO !
```

```
console.log(msg); // NO ! ASYNC !
```

```
fs.writeSync(process.stdout.fd, msg); // doc: SYNC !
```

```
process._rawDebug(msg); // in replace ! SYNC !
```

```
// Track from Pointer function
// have to make own wraper
// to Start Tracking

fn = () => { };
```

ONE FUNCTION EXPLANATION

SAME NUMBER OF ASYNC OP 4 PARALLEL SYNC CODE

```
fn = () => {                        // context mess :
    other_fn = () => {}             //      same asyncId
    other_fn = () => {}             //      same asyncId
    other_fn = () => {}             //      same asyncId
};
```

```
    TCPSERVERWRAP(5): trigger: 1 execution: 1
    TickObject(6): trigger: 5 execution: 1
    before:  6
      Timeout(7): trigger: 6 execution: 6
    after:   6
    destroy: 6
    before:  7
    >>> 7
      TickObject(8): trigger: 7 execution: 7
    after:   7
    before:  8
    after:   8
```

ONE FUNCTION EXPLANATION

ESCAPE ANALYSIS

https://en.wikipedia.org/wiki/Escape_analysis

SHAPE ANALYSIS (PROGRAM ANALYSIS)

https://en.wikipedia.org/wiki/Shape_analysis_(program_analysis)

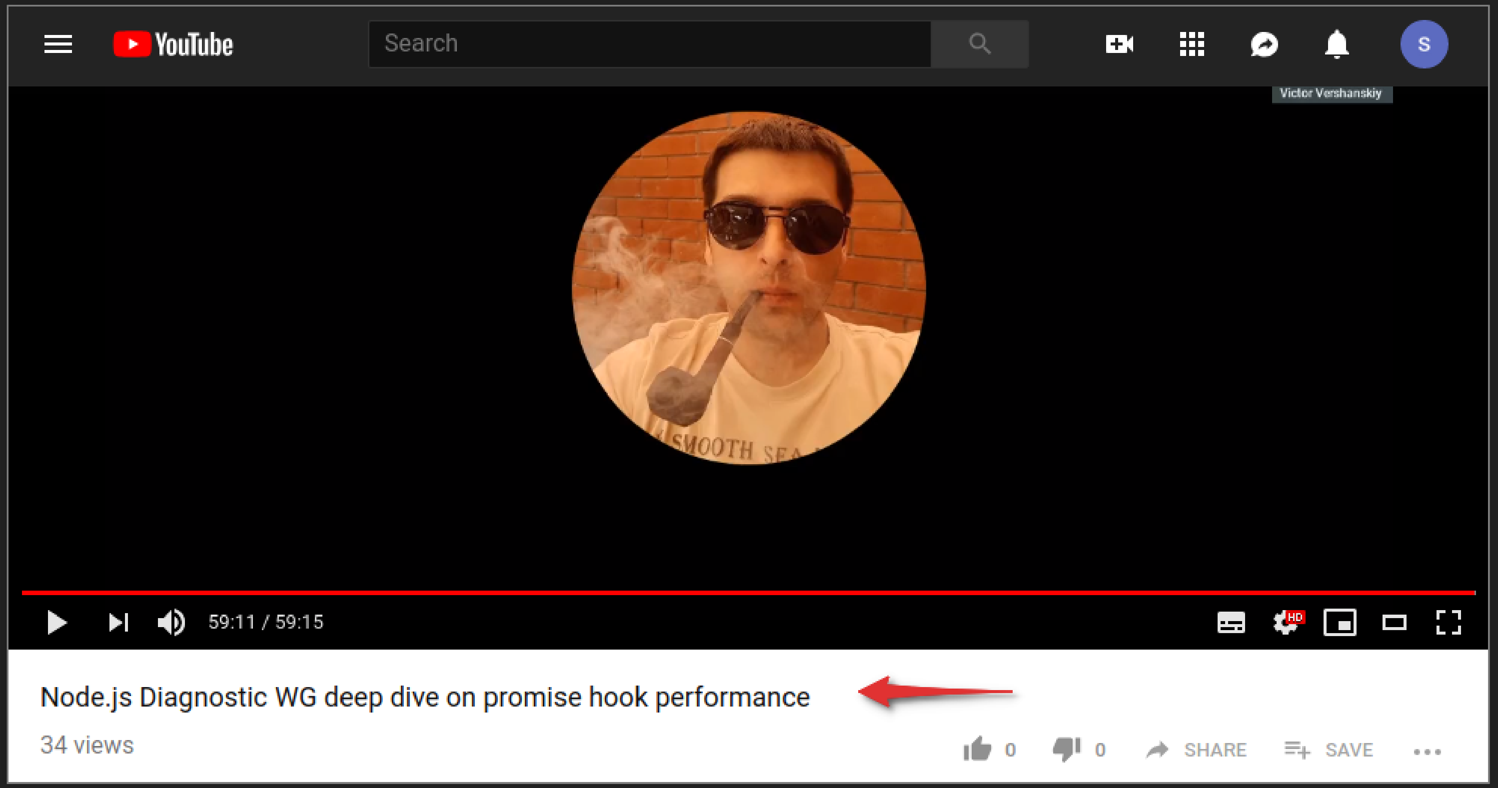
ALIAS ANALYSIS

https://en.wikipedia.org/wiki/Alias_analysis

PROMISES : ISSUE # 248

PERFOMANCE IS NOT SO GOOD

BUT THEY ARE FIXING



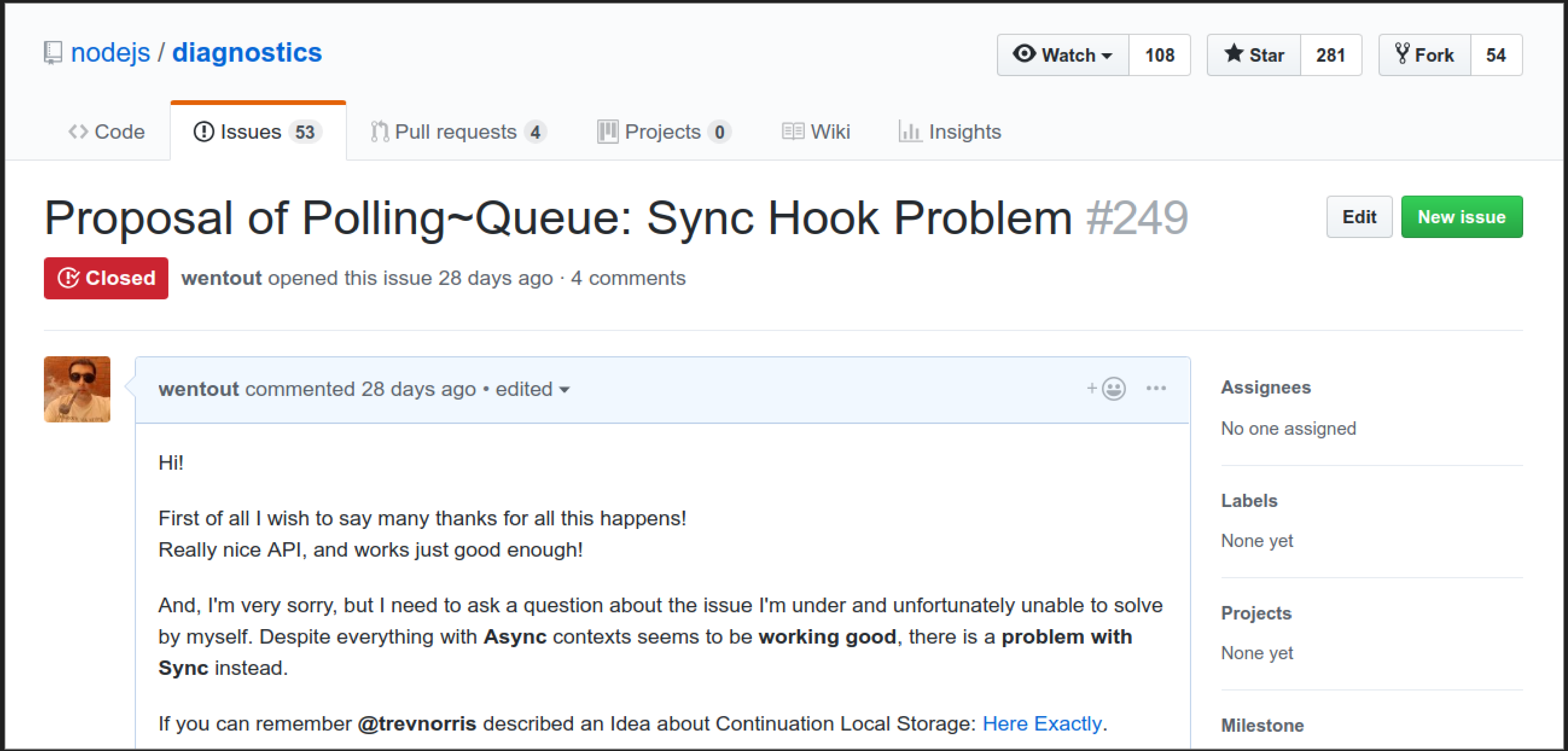
THE PROBLEM

```
const queueArray = [];

setInterval(() => {
    queueArray.forEach(task => {
        task();
    });
}, 1000);
```

```
const FunctionWhereTrackingStarts = () => {
    // Here we are still in the same context
    queueArray.push(() => {
        // **HERE** we have no ability
        // to reach the same logical context
        // because it executes in the other scope
    });
};
```

https://jsfiddle.net/evq7ht05/7/

MY ISSUE 249 STORY



nodejs / **diagnostics**

Watch 108 | Star 281 | Fork 54

Code | Issues 53 | Pull requests 4 | Projects 0 | Wiki | Insights

Proposal of Polling~Queue: Sync Hook Problem #249

Edit | New issue

Closed **wentout** opened this issue 28 days ago · 4 comments

wentout commented 28 days ago • edited

Hi!

First of all I wish to say many thanks for all this happens!
Really nice API, and works just good enough!

And, I'm very sorry, but I need to ask a question about the issue I'm under and unfortunately unable to solve by myself. Despite everything with **Async** contexts seems to be **working good**, there is a **problem with Sync** instead.

If you can remember **@trevnorris** described an Idea about Continuation Local Storage: Here Exactly.

Assignees

No one assigned

Labels

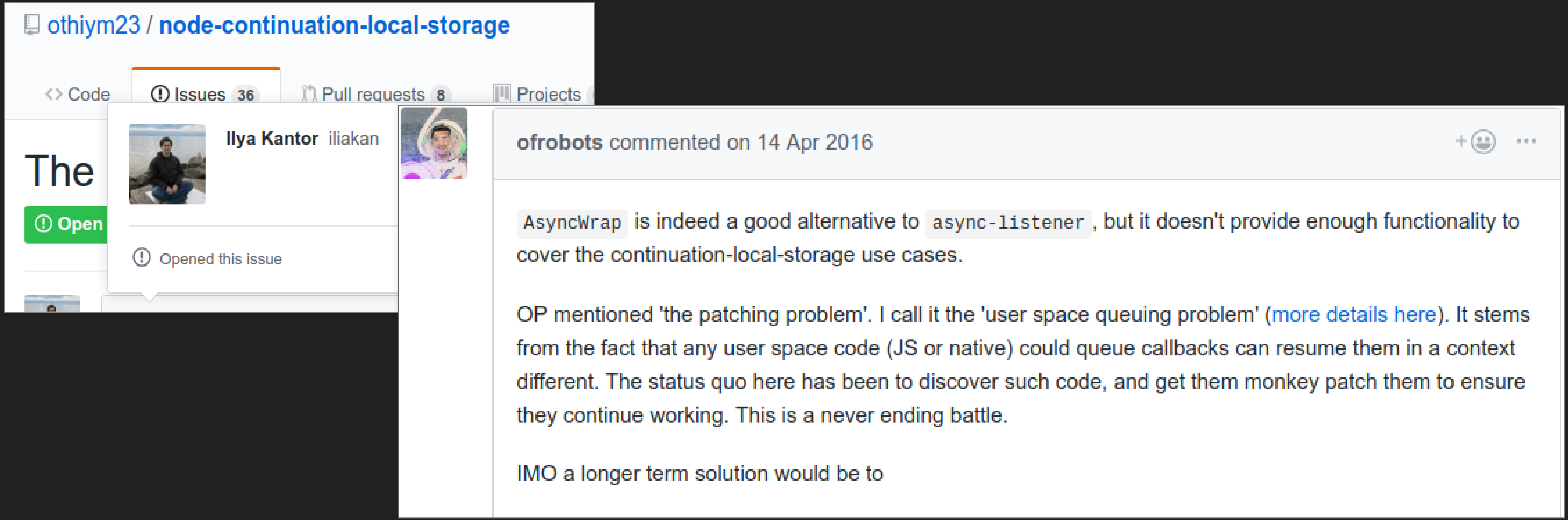
None yet

Projects

None yet

Milestone

ISSUE 59 : ILYA KANTOR



ASYNC CONTINUATION : 2015

Async context propagation in various language ecosystems

ofrobots@google.com, mattloring@google.com, Dec 30, 2015

Background

Async context propagation is the problem of being able to connect an async task to the originating "context" that queued/requested the async task in the first place. This is a problem that shows up in various async programming environments. This problem is present in JavaScript both on the client and the server side and there are several, similar but incompatible, mechanisms that have evolved in different communities to deal with it.

ASYNC CONTINUATION : 2017

Semantics of Asynchronous JavaScript

Matthew C. Loring
Google Inc, USA
mattloring@google.com

Mark Marron
Microsoft Research, USA
marron@microsoft.com

Daan Leijen
Microsoft Research, USA
daan@microsoft.com

Abstract

The Node.js runtime has become a major platform for developers building cloud, mobile, or IoT applications using JavaScript. Since the JavaScript language is single threaded, Node.js programs must make use of asynchronous callbacks and event loops managed by the runtime to ensure applications remain responsive. While conceptually simple, this programming model contains numerous subtleties and behaviors that are defined implicitly by the current Node.js implementation. This paper presents the first comprehensive formalization of the Node.js asynchronous execution model and defines a high-level notion of *async-contexts* to formalize fundamental relationships between asynchronous events in an application. These formalizations provide a foundation

asynchronous API's exposed by Node.js and for tools that can help develop [15], debug [4], or monitor [10, 24] the asynchronous behavior of their applications.

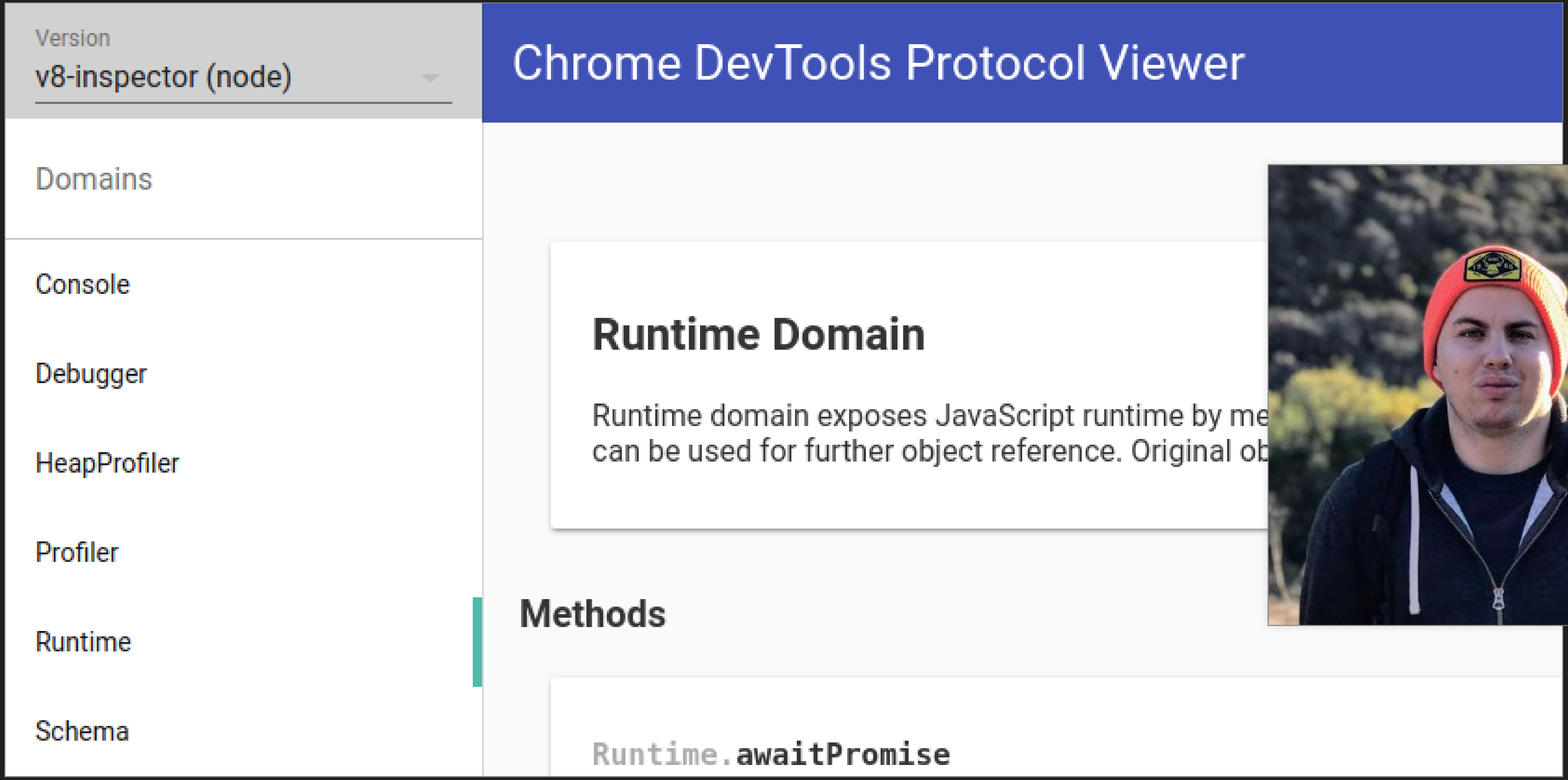
1.1 Semantics of Node Event Queues

A major challenge for research and tooling development for Node.js is the lack of a formal specification of the Node.js asynchronous execution model. This execution model involves multiple event queues, some implemented in the native C++ runtime, others in the Node.js standard library API bindings, and still others defined by the JavaScript *ES6 promise* language feature. These queues have different rules regarding when they are processed, how processing is interleaved, and how/where new events are added to each queue.

FIXING

6

CHROME DEBUG PROTOCOL : EXAMPLES



NO WAY EVENT TRACING INTERCEPTION

LIVE DEMOS!

-- trace flag : test.js

%DebugTrace(); : testDT.js

ASKING ANY HELP

Anna Henningsen
Andreas Madsen
Tomas Watson

Async Hooks CLS review request _context-dive ×

system
Hello Andreas, I'm sorry for writing this email, just got it from github and so on. Seems you ar

Andreas Madsen
to me

Hi

Sorry I haven't answered earlier. Yeah, I'm a bit stressed at the moment as I'm trying to finish so

· you are saving the resource object in your "it" object, which will per

solution.

Kind Regards

Really Sorry for interrupting

Anna
@addaleax
queer. she/her. non-binary tarball, tabs woman, unicorn-coloured hair.
Düsseldorf, Germany | @NearFor

this looks like exactly the kind of problem solved by nodejs.org/api/async_hook...
Oct 20

Hi!

Extremely great you answe

question about async_hooks

system
Hi Tomas! Sorry for interruping! Nice to meet you

Thomas Watson
to me

If I understand your proposal correctly, you are adv

JavaScript Embedder API

Library developers that handle their own asynchronous resources performing tasks like I/O, connection pooling, or managing callback queues may use the `AsyncWrap` JavaScript API so that all the appropriate callbacks are called.

Class: AsyncResource [src]

The class `AsyncResource` is designed to be extended by the embedder's async resources. Using this, users can easily trigger the lifetime events of their own resources.

The `init` hook will trigger when an `AsyncResource` is instantiated.

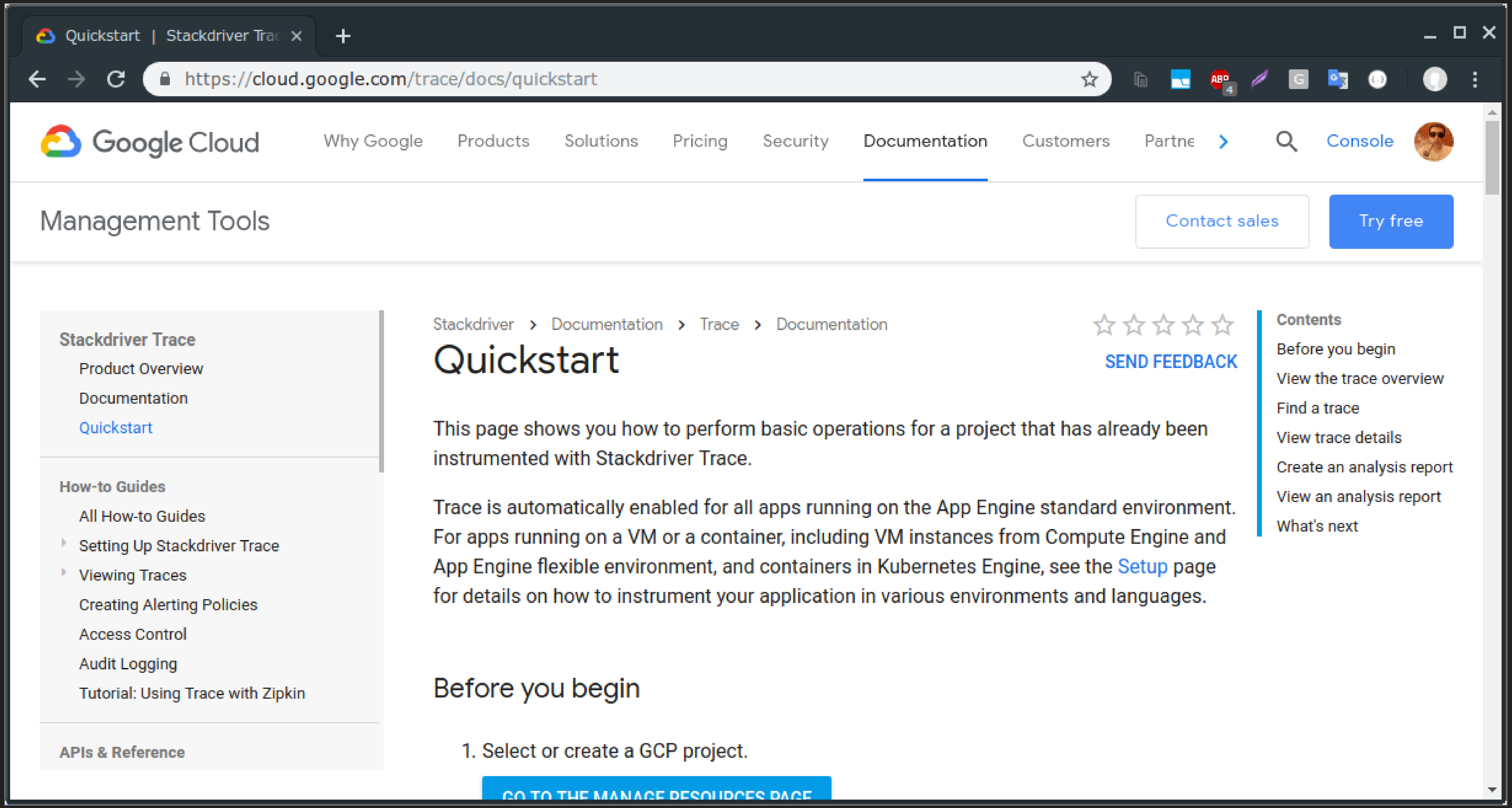
SOLUTION

WORKING CONCEPT IMPL

```
function (fn2wrap) {
    if (context) {
        fn2wrap = dive(fn2wrap, context);
        return fn2wrap;
    }
}
```

```
if (context) {
    args = args.map(arg => {
        if (typeof arg === 'function') {
            arg = dive(arg, context);
        }
        return arg;
    });
}
```

CLOUD TRACE



Quickstart | Stackdriver Trac
https://cloud.google.com/trace/docs/quickstart
Google Cloud
Why Google
Products
Solutions
Pricing
Security
Documentation
Customers
Partne
Console
Management Tools
Contact sales
Try free
Stackdriver Trace
Product Overview
Documentation
Quickstart
How-to Guides
All How-to Guides
Setting Up Stackdriver Trace
Viewing Traces
Creating Alerting Policies
Access Control
Audit Logging
Tutorial: Using Trace with Zipkin
APIs & Reference
Stackdriver › Documentation › Trace › Documentation
Quickstart
SEND FEEDBACK
This page shows you how to perform basic operations for a project that has already been instrumented with Stackdriver Trace.
Trace is automatically enabled for all apps running on the App Engine standard environment. For apps running on a VM or a container, including VM instances from Compute Engine and App Engine flexible environment, and containers in Kubernetes Engine, see the Setup page for details on how to instrument your application in various environments and languages.
Before you begin
1. Select or create a GCP project.
Contents
Before you begin
View the trace overview
Find a trace
View trace details
Create an analysis report
View an analysis report
What's next

CLOUD TRACE

https://github.com/googleapis/cloud-trace-nodejs/blob/master/src/plugins/plugin-mongodb-core.ts

```
    patch: function(mongo, api) {
      shimmer.wrap(mongo.Server.prototype, 'command', wrapWithLabel(api, 'mongo-command'));
      shimmer.wrap(mongo.Server.prototype, 'insert', wrapWithLabel(api, 'mongo-insert'));
      shimmer.wrap(mongo.Server.prototype, 'update', wrapWithLabel(api, 'mongo-update'));
      shimmer.wrap(mongo.Server.prototype, 'remove', wrapWithLabel(api, 'mongo-remove'));
      shimmer.wrap(mongo.Cursor.prototype, 'next', createNextWrap(api));
    },
    unpatch: function(mongo) {
      shimmer.unwrap(mongo.Server.prototype, 'command');
      shimmer.unwrap(mongo.Server.prototype, 'insert');
      shimmer.unwrap(mongo.Server.prototype, 'update');
      shimmer.unwrap(mongo.Server.prototype, 'remove');
      shimmer.unwrap(mongo.Cursor.prototype, 'next');
    }
```

LIVE DEMOS!

mongoose polling jump

async code branch jump

RESUME

PROS

- LTS works
- tracking
- async ()

CONS

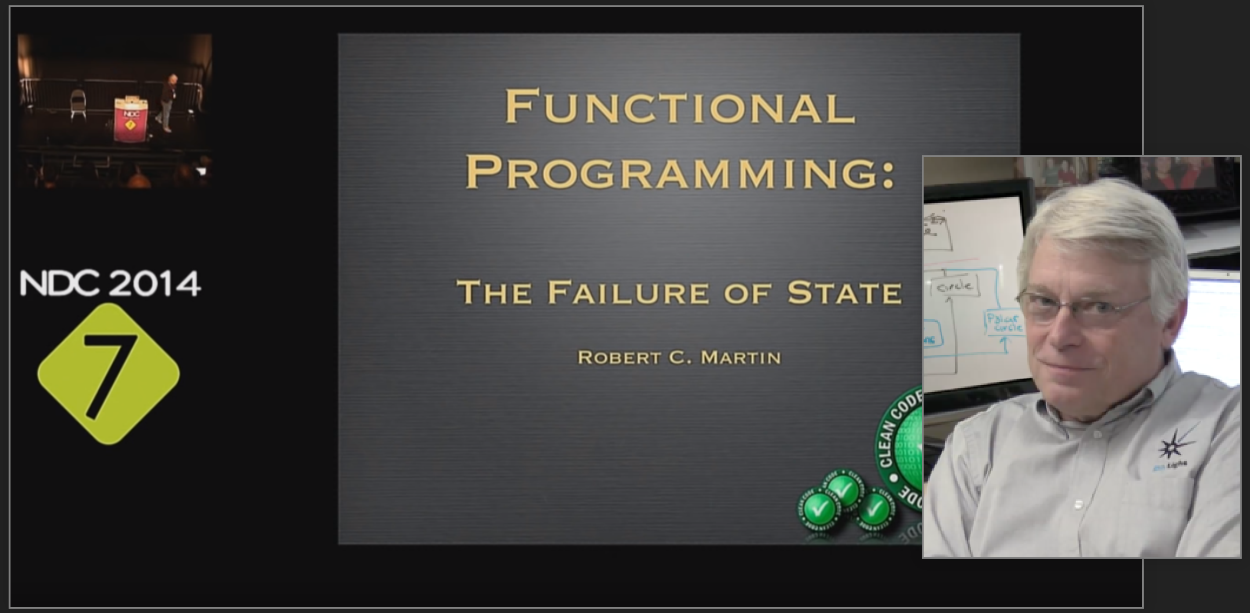
- ! prod ready yet
- perfomance /3
- cls stops gc

CLS NPM

cls-hooked
continuation-local-storage

and — many — many — other packages

CONTEXT DISCUSSIONS



AND MAIN IDEA

!== BEING SHY TO ASK

AND COMMUNICATE

THE END

I know how to make it faster
... but it would be slower ...

NODE CREW THANK YOU !

for giving and who works with

async_hooks

AND SPECIAL THANKS TO

- Aleksey Kozyatinskiy for giving a way
- Thomas Watson for patient explanations
- Andreas Madsen for willing to help
- Anna Henningsen for answering at all
- Trevor Norris for the begining
- Thorsten Lorenz for docs & exampes
- Ilya Kantor for issue #59

TEAM THANK YOU !

Stickers Pack : uCircus

ALL IN GOOD TIMES

MY GIT ~ @WENTOUT

http://context-dive.com/holy

made using reveal.js & plugins, thanks!