

uikit

# ASYNCHRONOUS ZERO

# WHO AM I

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

nowdays for [ukit.group](https://www.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.

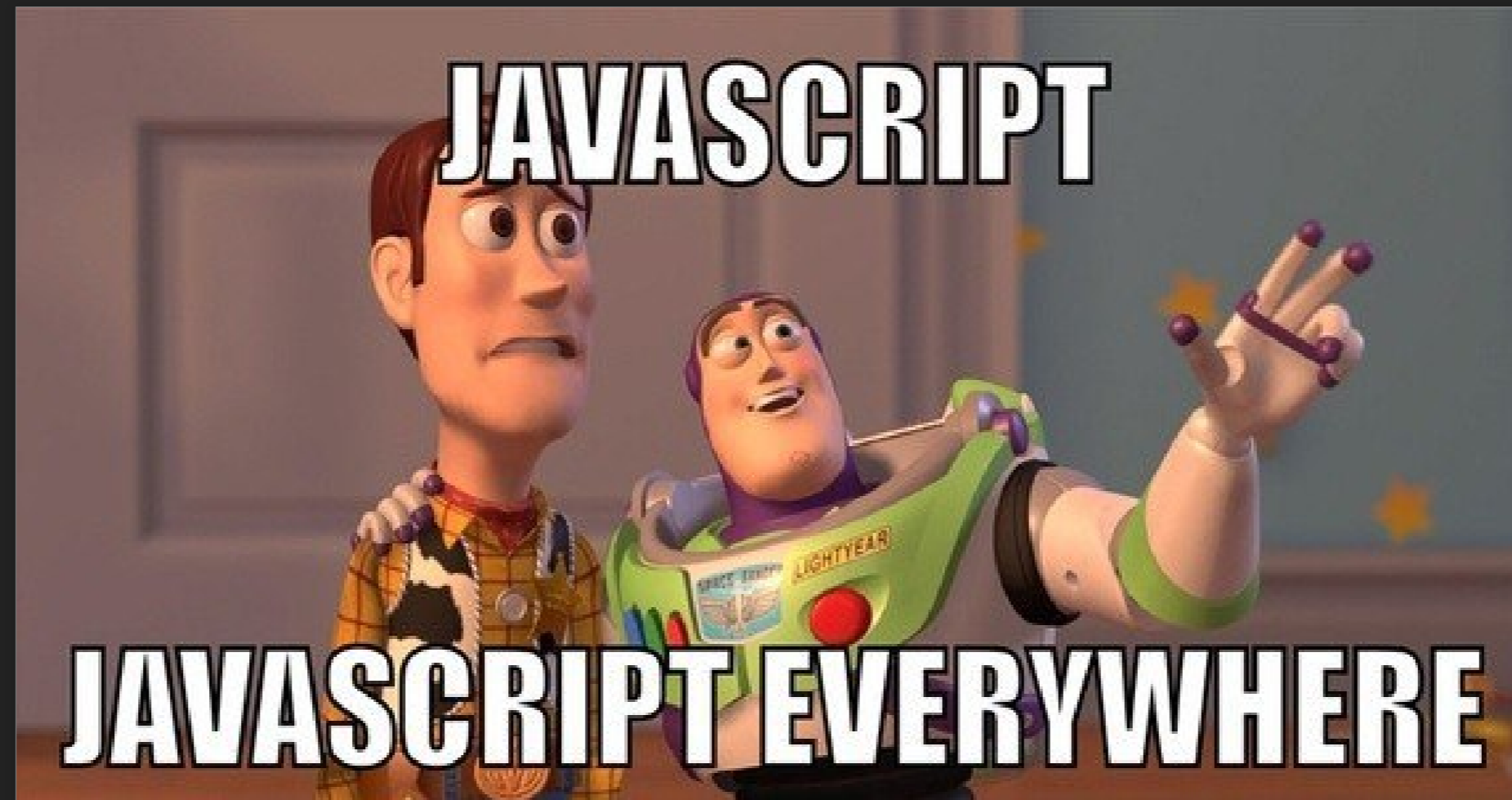
Then XUL Runner. Cordova. ExtJS.

**AND FINALLY NODEJS.**

meanwhile 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*

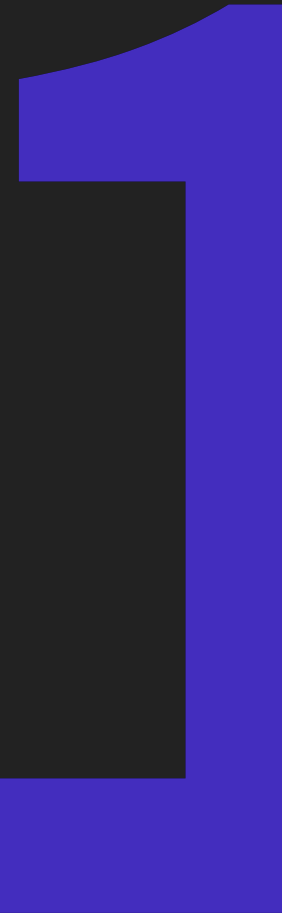
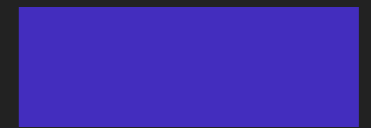
# SO THIS TALK WAS MADE

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





... **INTRO** ...





**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
```

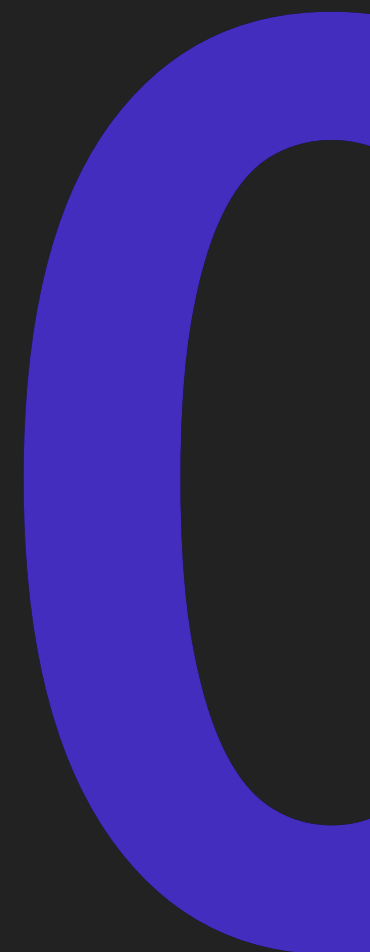
```
// & for bugs
```

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



MOSCOW  
2017

**Николай Матвиенко**

Grid Dynamics

Поиск и устранение  
неисправностей  
Node.js-приложений  
под капотом



# DEBUGGING IN PRODUCTION

**HolyJS** // Москва 2016

---

Thomas Watson

Opbeat

Debugging Node.js  
Performance Issues  
in Production



# NODE FATAL FLOW



MOSCOW  
2017

Алексей Охрименко

IRONWEB

Фатальный  
недостаток Node.js





# WHAT WE HAVE

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







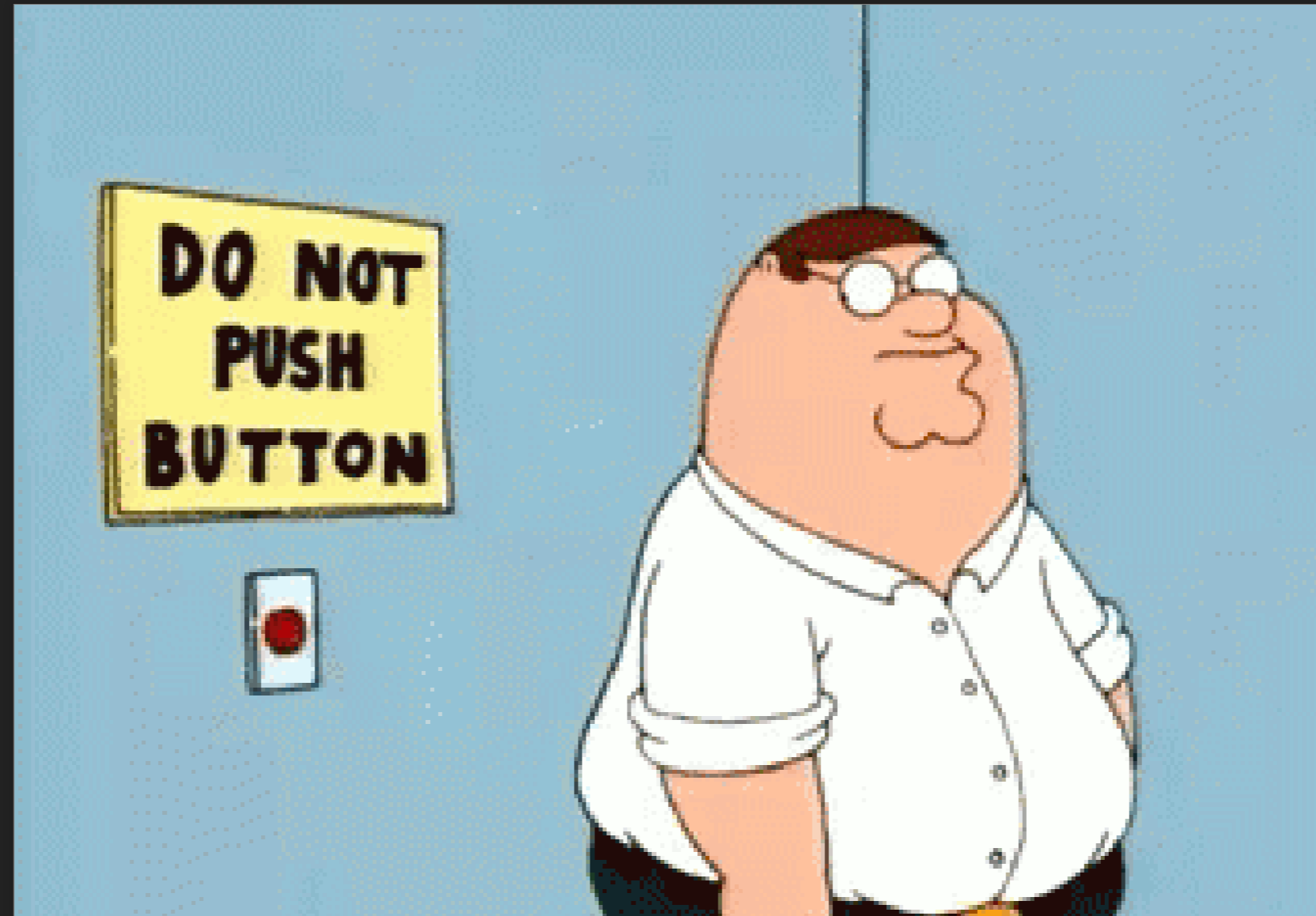
# metaphor

OF ASYNC

# 4 CODING



— AND SOMEHOW ... SOMEWHERE —



# SOMEONE ASSIGNS ME A BUG

Issue Type\*  ?

Some issue types are unavailable due to incompatible fields.

Main **Additional**

Summary\*

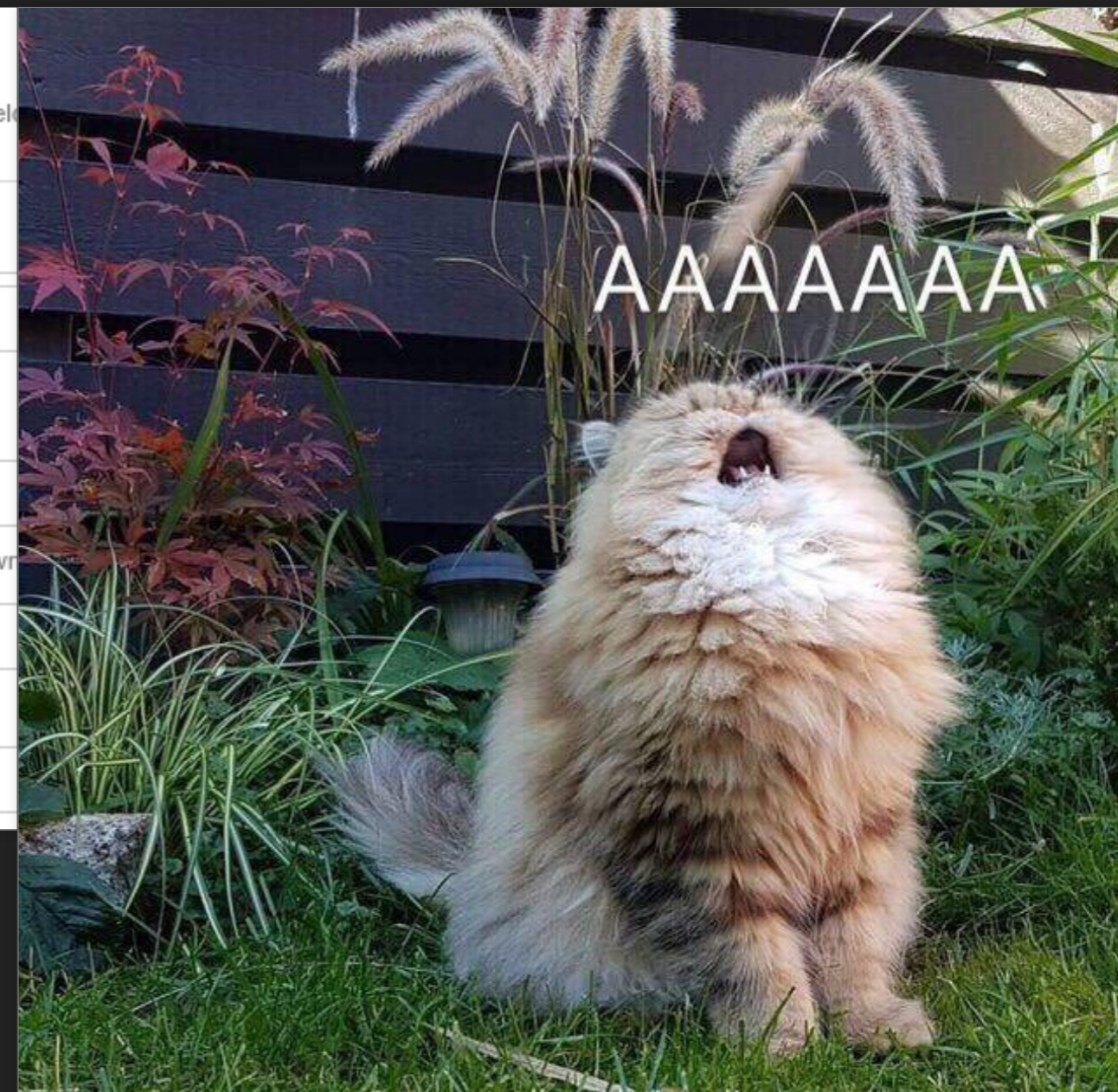
Priority  ?

Component/s

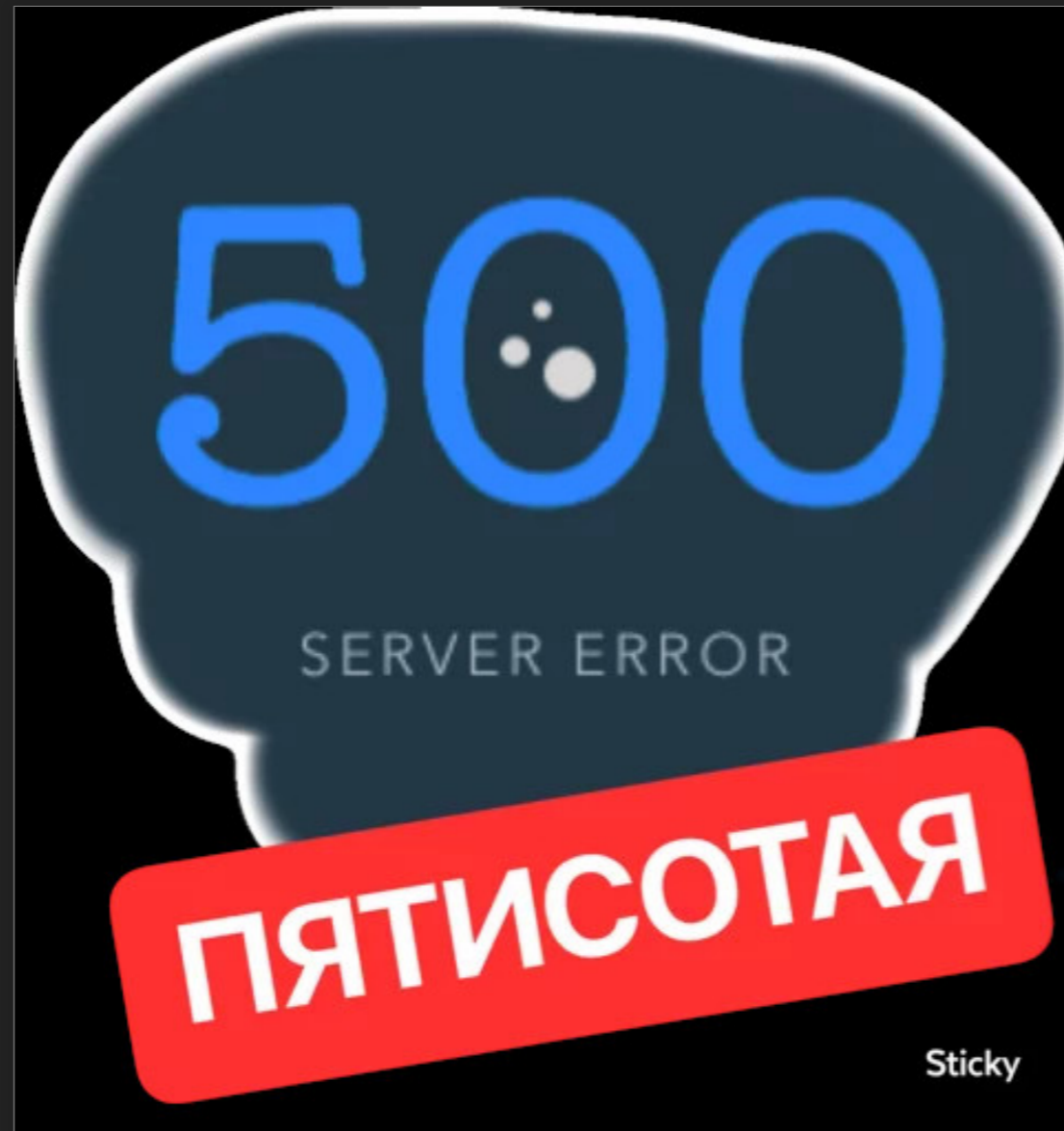
Assignee

Story Points

- Нормально
- Очень срочно
- Очень важно
- Важно
- Когда-нибудь
- Маловажное возможное улуч...



and it that situation



```

45|queue_s | | error:
45|queue_s | | message: [undefined]
45|queue_s | | data:
45|queue_s | | id: 26●●●7
45|queue_s | | domain: d●●●●●●●y.com
45|queue_s | | ●●●●●●●●
45|queue_s | | expiration_date: 2019-11-14T10:41:19Z
45|queue_s | | _uuid: 5b●●●●f
45|queue_s | | userDomainName: d●●●●●●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

```
45|queue_s | error:
45|queue_s | message: [undefined]
45|queue_s | data:
45|queue_s | id: 26●●●7
45|queue_s | domain: d●●●●●●●y.c
45|queue_s | expiration_date: 2019-11
45|queue_s | uuid: 5b●●●●●f
```

```
Message from syslogd@ul2 at Jan 19 12:19:51 ...
kernel:[30582052.591040] NMI watchdog: BUG: soft lockup - CPU#10 stuck for 22s! [pidof:8724]

Message from syslogd@ul2 at Jan 19 12:19:54 ...
kernel:[30582055.947062] NMI watchdog: BUG: soft lockup - CPU#2 stuck for 22s! [pidof:9049]
```

```
1 [|||||] 100.0%
2 [|||||] 100.0%
3 [|||||] 100.0%
4 [|||||] 100.0%
Mem [|||||] 29.1G/62.9G
Tasks: 420; 9 running
```

```
at process.nextTick (/home
at args.(anonymous functio
at process._tickCallback (
2018-11-14 13:41 +03:00: 11.
error:
Error: Callback was already
stack:
/home/www/ulight/node_mo
/home/www/ulight/node_mo
/home/www/ulight/node_mo
/home/www/ulight/node_mo
model.$_save.error (/ho
/home/www/ulight/node_mo
next (/home/www/ulight/n
/home/www/ulight/node_mo
process.nextTick (/home/
args.(anonymous function
process._tickCallback (i
```

```
20.0
10.0
14:20
— alarm — load 1
```

```
Message from syslogd@ul2 at Jan 19 12:20:11 ...
kernel:[30582072.191175] NMI watchdog: BUG: soft lockup - CPU#5 stuck for 23s! [pidof:7869]

Message from syslogd@ul2 at Jan 19 12:20:11 ...
kernel:[30582072.351177] NMI watchdog: BUG: soft lockup - CPU#7 stuck for 22s! [pidof:7465]

Message from syslogd@ul2 at Jan 19 12:20:11 ...
kernel:[30582072.831181] NMI watchdog: BUG: soft lockup - CPU#13 stuck for 22s! [pidof:7892]

Message from syslogd@ul2 at Jan 19 12:20:11 ...
kernel:[30582072.911180] NMI watchdog: BUG: soft lockup - CPU#14 stuck for 22s! [zabbix_agentd:6350]
```





WHEN YOU HAVE STACK... BUT

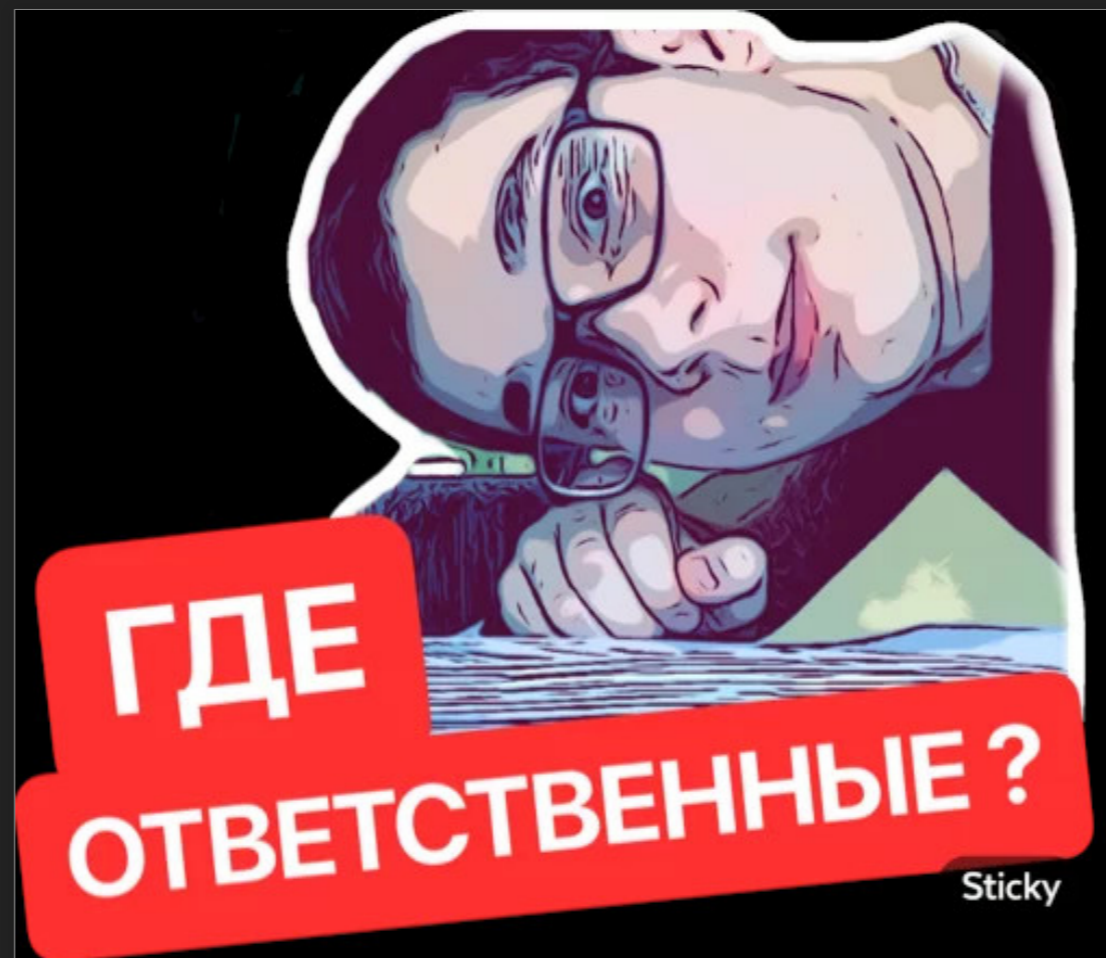
UNABABLE TO

```
45|queue_s|ror:
45|queue_s|message: [un...ed]
45|queue_s|data:
45|queue_s|id: 26...
45|queue_s|domain: d...
45|queue_s|expiration: 2019-1...
45|queue_s|uid: 5b...
45|queue_s|
45|queue_s|1|E|100.0%|
45|queue_s|2|E|100.0%|
45|queue_s|3|E|100.0%|
45|queue_s|4|E|100.0%|
45|queue_s|Mem|29.1G/62.9G| Tasks: 420; 9 running
45|queue_s|
45|queue_s|at process.nextTick (/home
45|queue_s|at args.(anonymous_functio
45|queue_s|at process.
45|queue_s|14 13:4
45|queue_s|ror: Call... was already
45|queue_s|k:
45|queue_s|home/www...
45|queue_s|/home/www...
45|queue_s|/home/www...nt/node_mo
45|queue_s|/home/www...ht/node_mo
45|queue_s|model.$...error (/ho
45|queue_s|/home/www...ht/node_mo
45|queue_s|ext (/ho.../light/s
45|queue_s|home/www
45|queue_s|process.nextTick (/home/
45|queue_s|args.(anonymous_functio
45|queue_s|process_tickCallback (i
Message from syslogd@ul2 at Jan 19 12:19:11 ...
kernel:[30582052.1040] NMI watchdog: BUG: soft lockup - CPU#10 stuck for 22s! [pidof:8
Message from syslogd@ul2 at Jan 19 12:19:11 ...
kernel:[30582055.2062] NMI watchdog: BUG: soft lockup - CPU#2 stuck for 22s! [pidof:90
Message from syslogd@ul2 at Jan 19 12:20:11 ...
kernel:[30582072.351177] NMI watchdog: BUG: soft lockup - CPU#14 stuck for 22s! [pidof:7
Message from syslogd@ul2 at Jan 19 12:20:11 ...
kernel:[30582083.11817] NMI watchdog: BUG: soft lockup - CPU#14 stuck for 22s! [pidof:2] [INT]
Message from syslogd@ul2 at Jan 19 12:20:11 ...
kernel:[30582072.911180] NMI watchdog: BUG: soft lockup - CPU#14 stuck for 22s! [zabbix_agentd:6350]
```

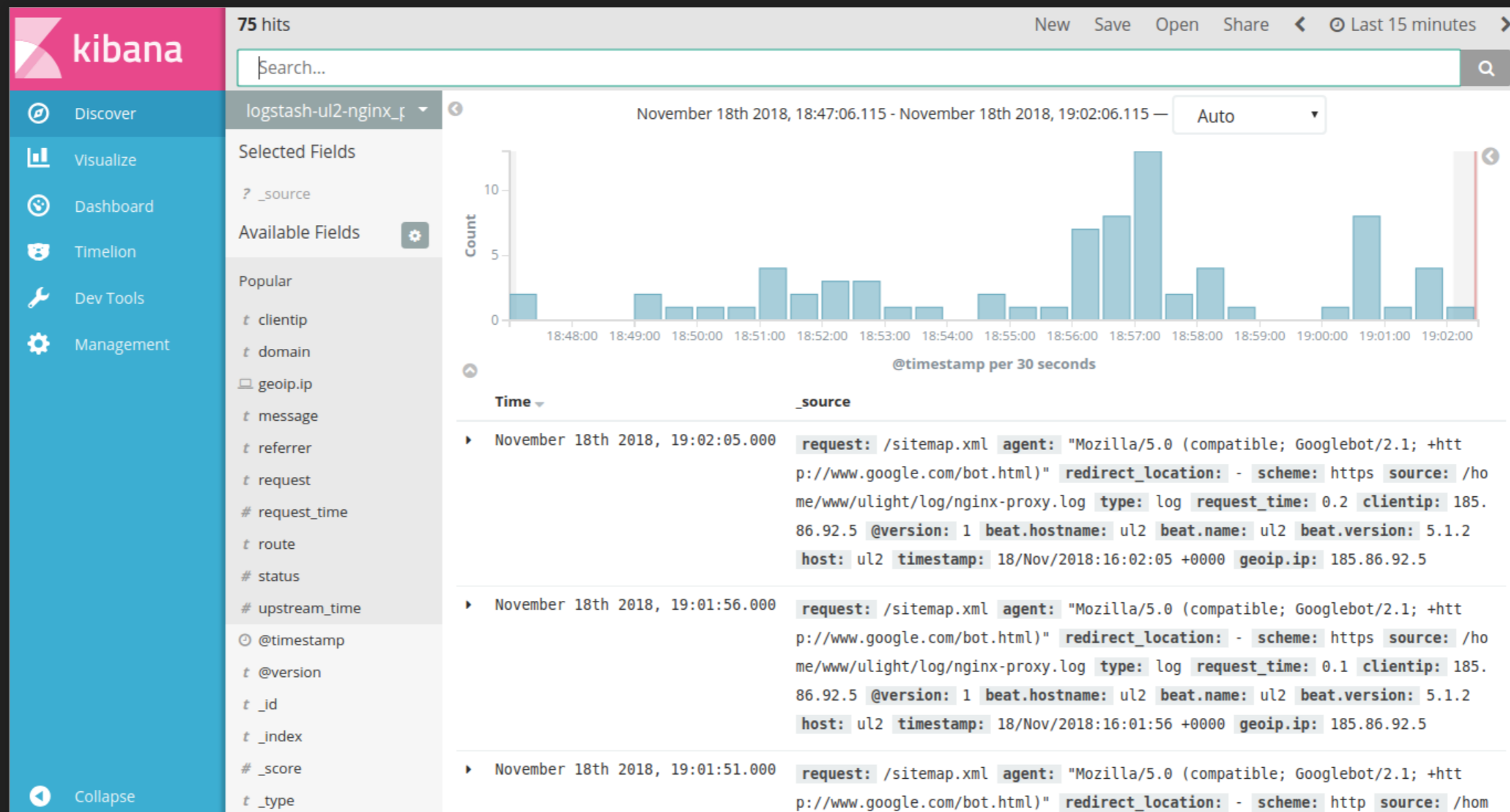
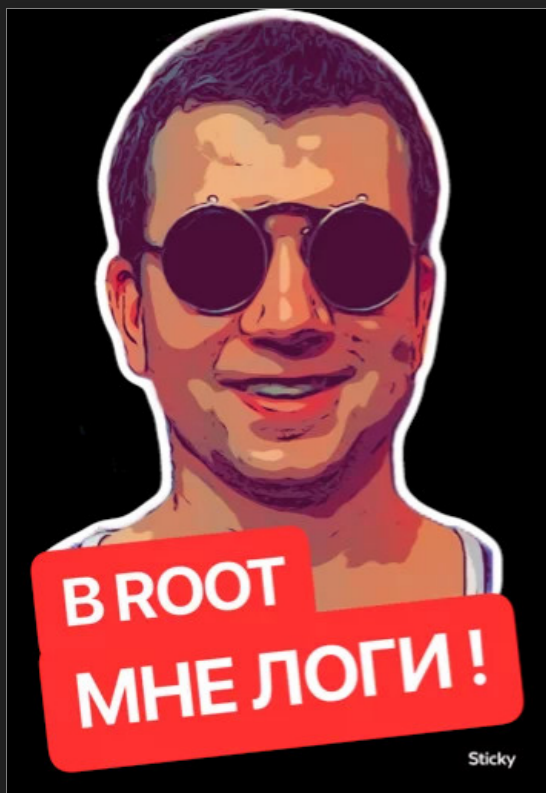
REPRODUCE



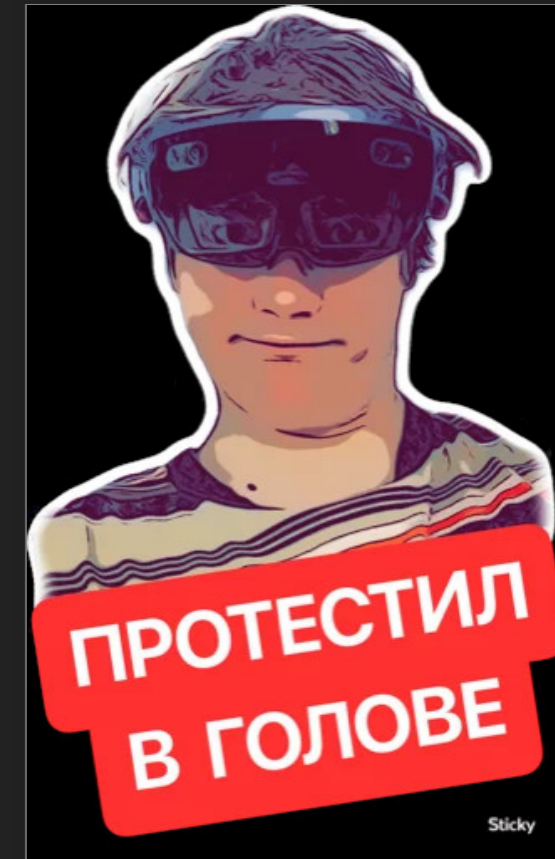
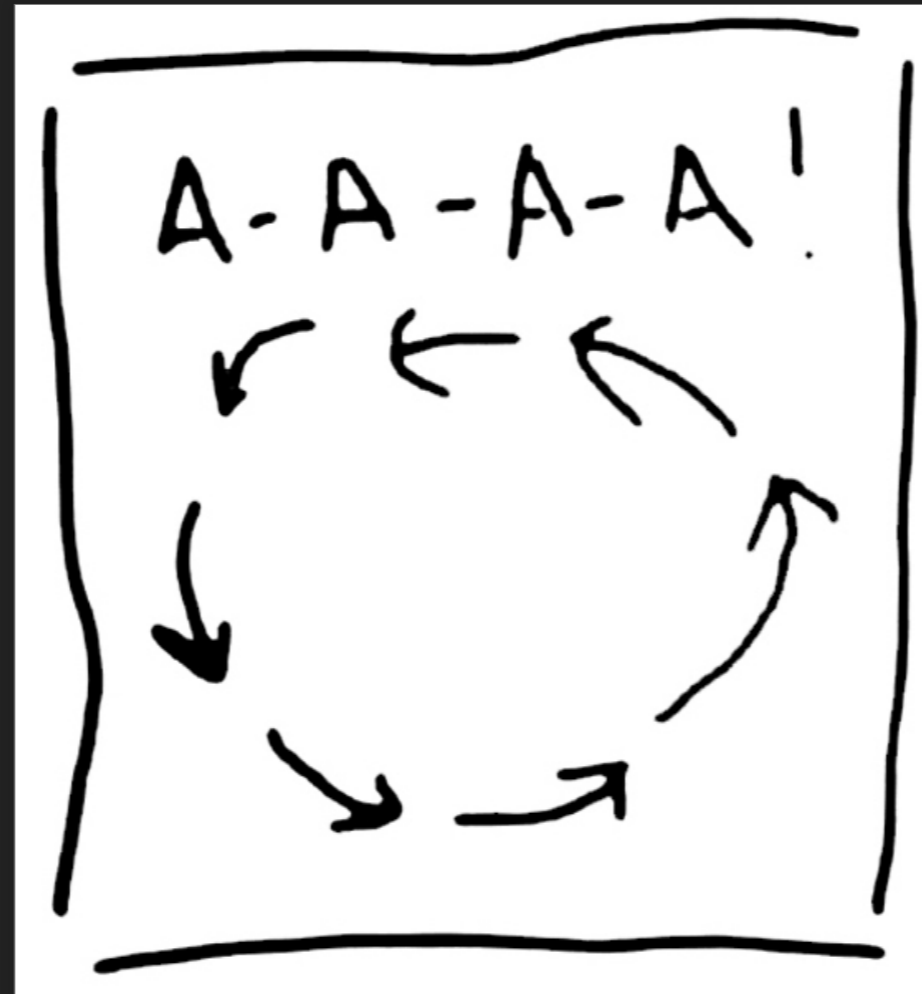
# BOSS



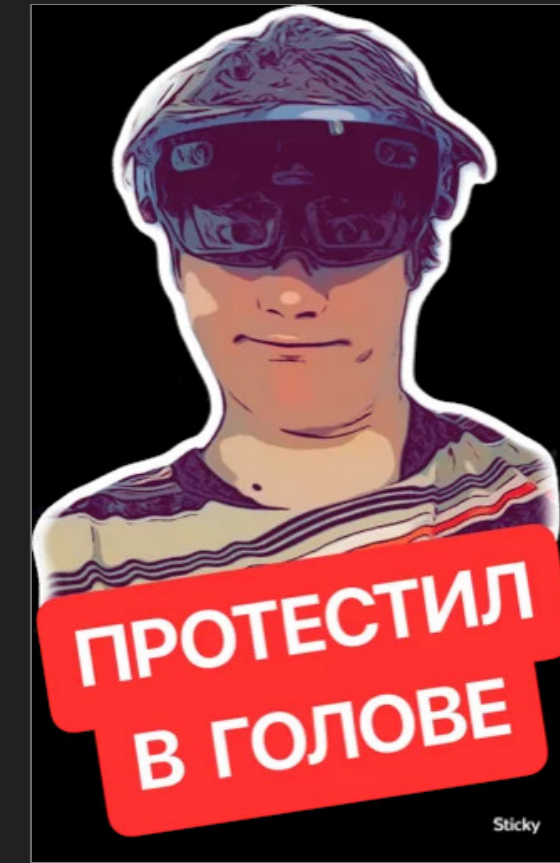
# YOU COLLECT ALL AVAILABLE DATA



# YOU COLLECT ALL AVAILABLE DATA PREPEARE TEST SAMPLES



YOU COLLECT ALL AVAILABLE DATA  
PREPEARE TEST SAMPLES  
NOTHING HELPS

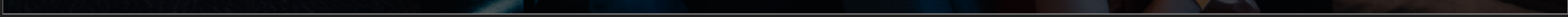


SO WHAT IS IT ?  
**SCHRODINGER'S**  
**CODE ?**



**HEISENBUG**

December 6-7, 2018





# LIVE DEMO !

what the 500 looks like : app.js

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

**WE HAVE TRACE & CODE**

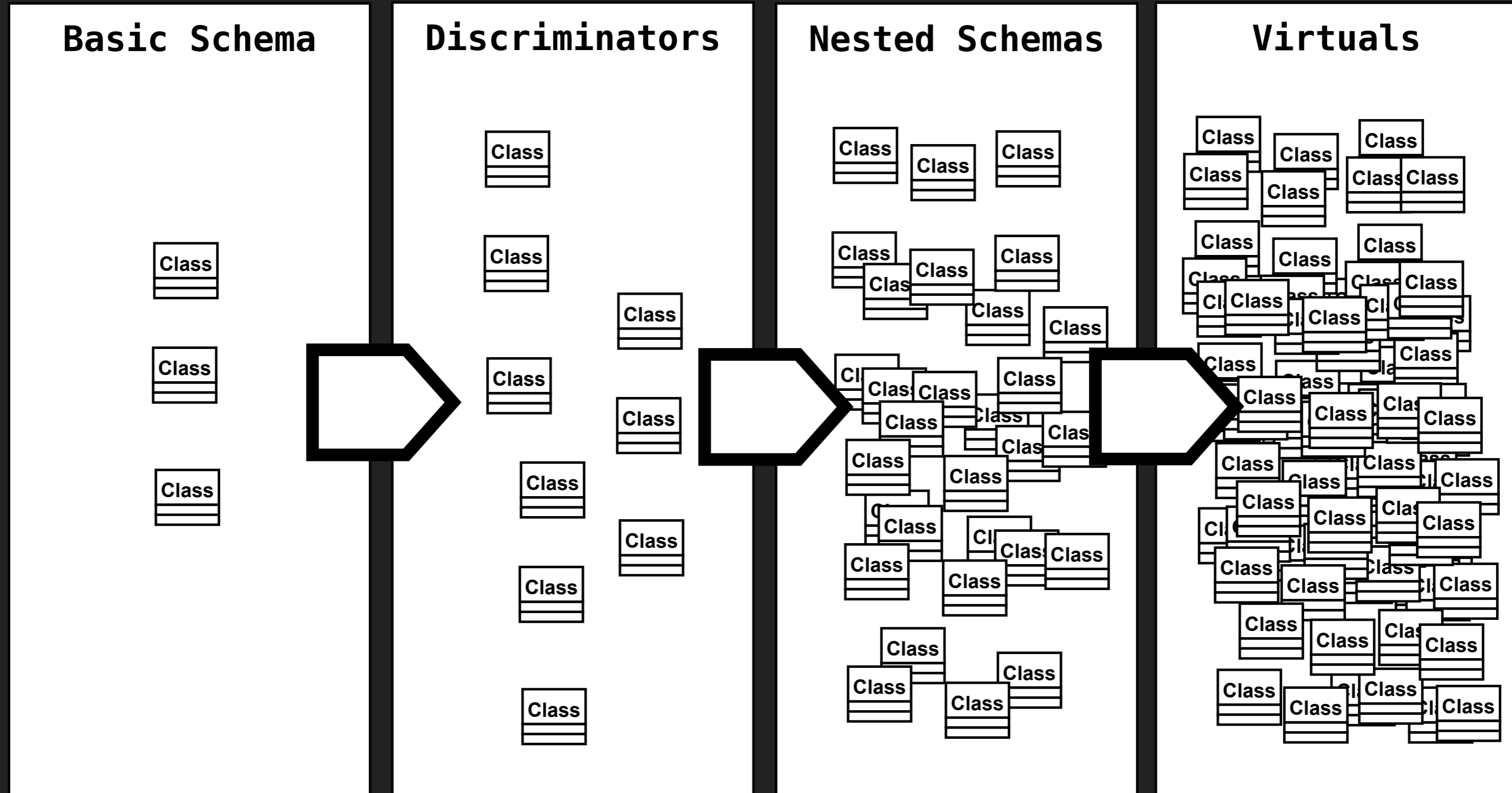
**BUT NOT**

**THE STATE 2 FIX**

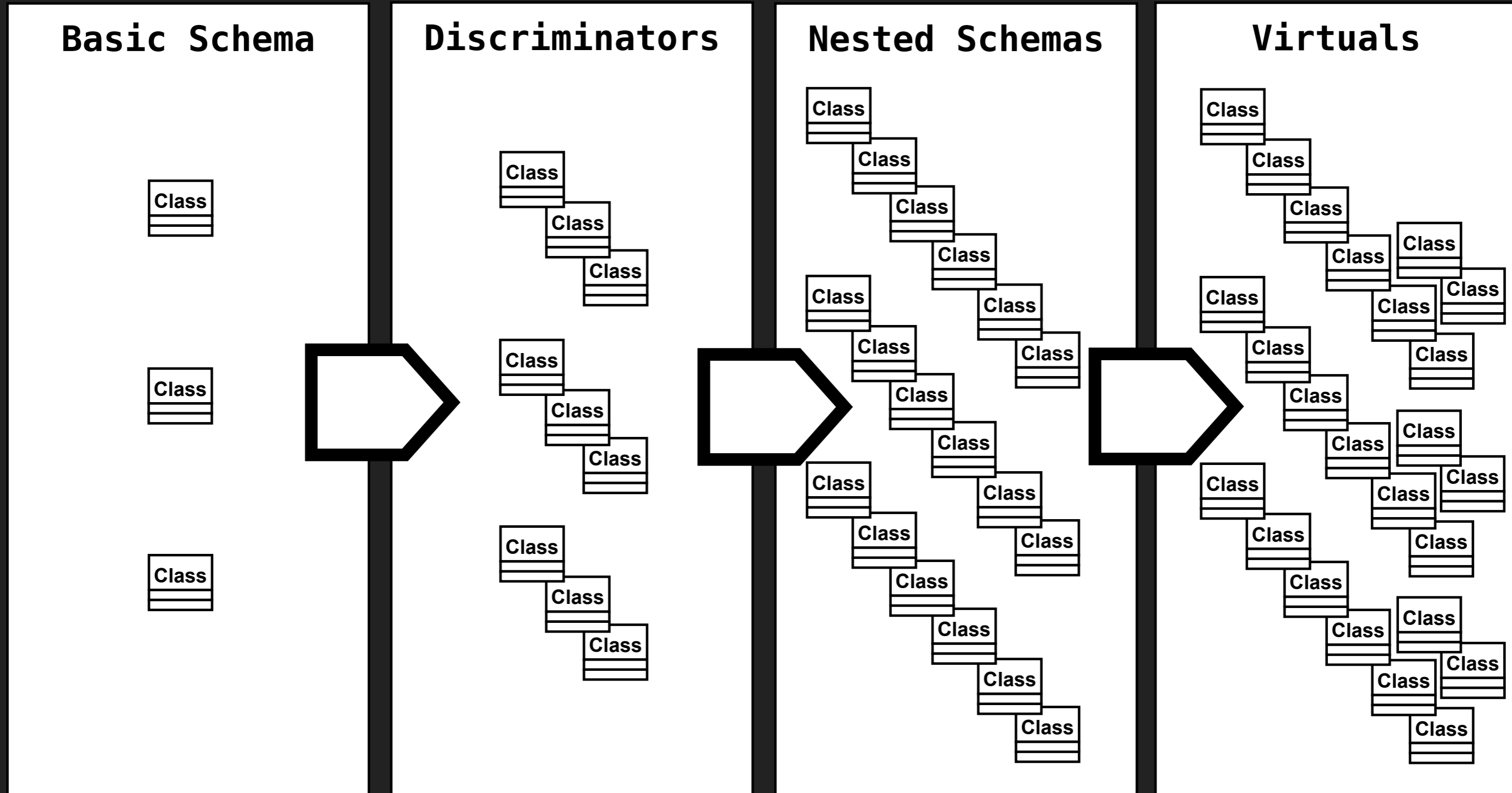
**MEANING : WHEN & HOW ?**

**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"} ⓘ
```

```
  foo: "bar"
```



## AND HAVE STATE DRIVEN ERRORS

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



using console.log

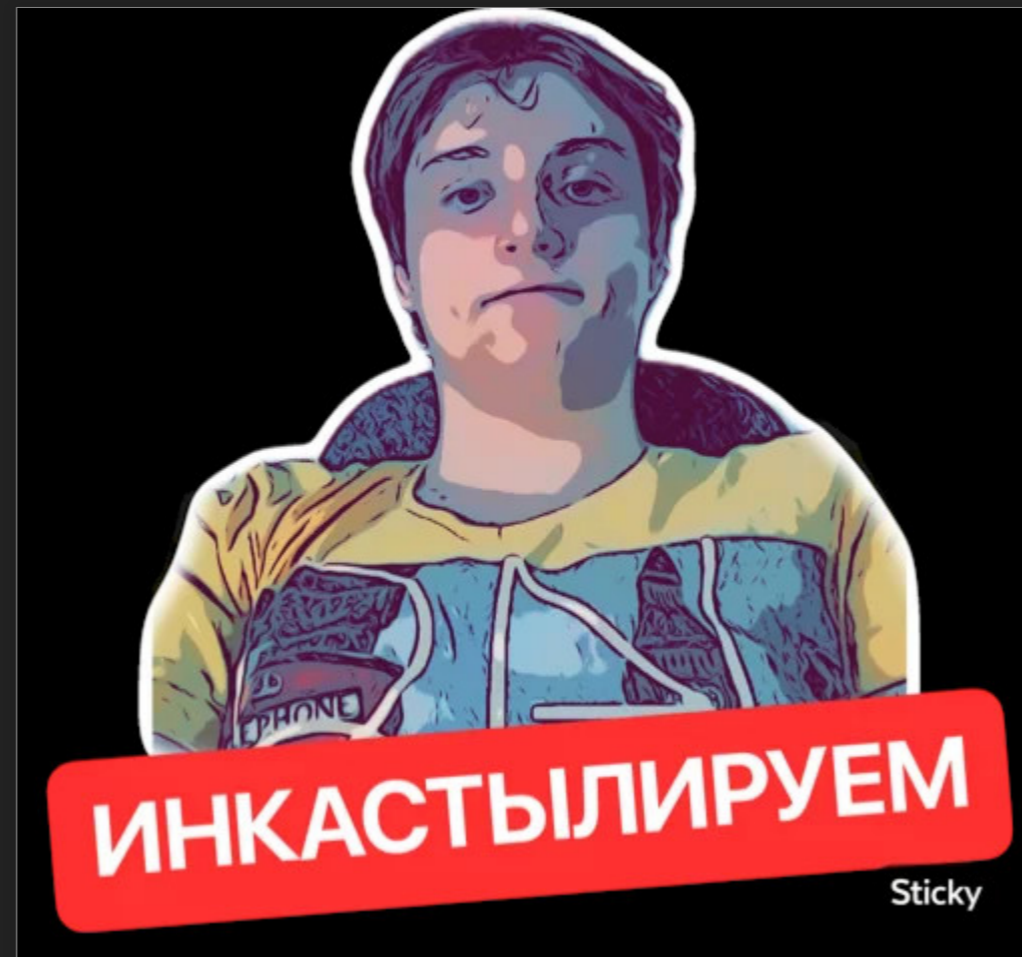
## WE OOPS : CHANGED BVR & STATE

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

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



# SHOULD MAKE A SPIKE





# DIVING

an approach  
hopeless though...





# 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 Verification

<https://ieeexplore.ieee.org/document/7546839/>

by Z Zhou - 2017 - Cited by 240 - Related articles

Geometric consistency verification is a popular technology for re  
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](#)*



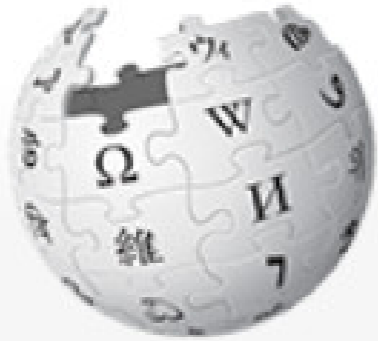
This article **needs additional citations for verification**. Please help improve it by adding citations to reliable sources. Unsourced material may be challenged and removed. *(June 2009)* *([Learn how and when to remove this message](#))*

In the context of hardware and software systems, **formal verification** is the act of proving or disproving the correctness of a system with respect to a property, using formal methods of mathematics.<sup>[1]</sup>

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 of the system, otherwise known by construction. Examples of mathematical objects often used to model systems are: finite state machines, hybrid automata, process algebra, formal semantics of programming languages such as operational semantics

# 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 with a proof, a result. It is called *efficient* if the combined runtime of the algorithm and a proof checker is slower by at most a constant factor than the algorithm alone.

The proof produced by a certifying algorithm should be in some sense simpler than the algorithm's output (e.g., the proof should be found in [linear time](#)) simplicity of the output proof is considered in a less formal sense.<sup>[1]</sup> For instance, a proof checker for the algorithm or a checker for the proof may be more amenable to [formal verification](#).<sup>[1][2]</sup>

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

# Graduate Student Solves Quantum Verification Problem

61 |

*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](https://en.wikipedia.org/wiki/Set_theory)

## **THEORY OF CONSTRAINTS**

[https://en.wikipedia.org/wiki/Theory\\_of\\_constraints](https://en.wikipedia.org/wiki/Theory_of_constraints)

## **SELF-CLOCKING SIGNAL**

[https://en.wikipedia.org/wiki/Self-clocking\\_signal](https://en.wikipedia.org/wiki/Self-clocking_signal)

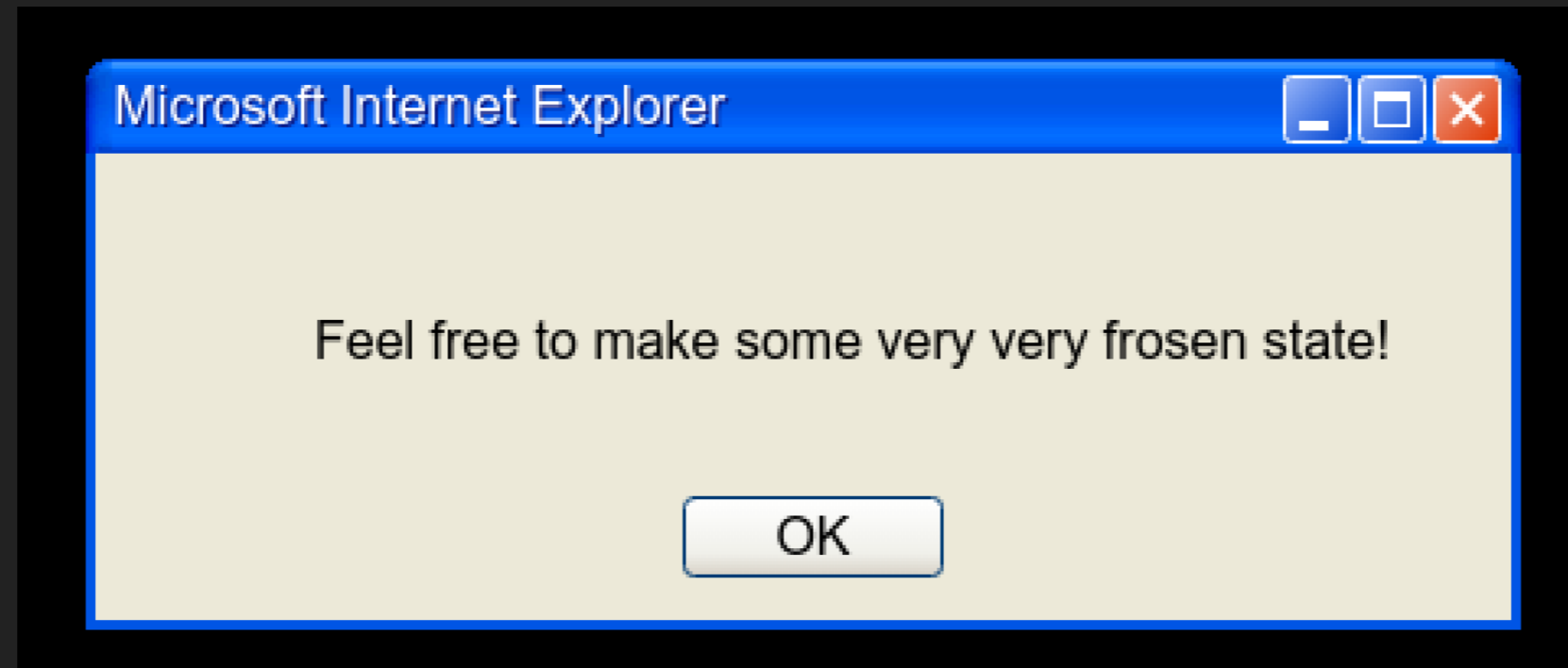
## **SELF-SYNCHRONIZING CODE**

[https://en.wikipedia.org/wiki/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 ?



## 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 ?

## Profiling Node.js

by Dave Pacheco, 2012-04-25

It's incredibly easy to visualize  
of Brendan Gregg's [FlameGraph](#)

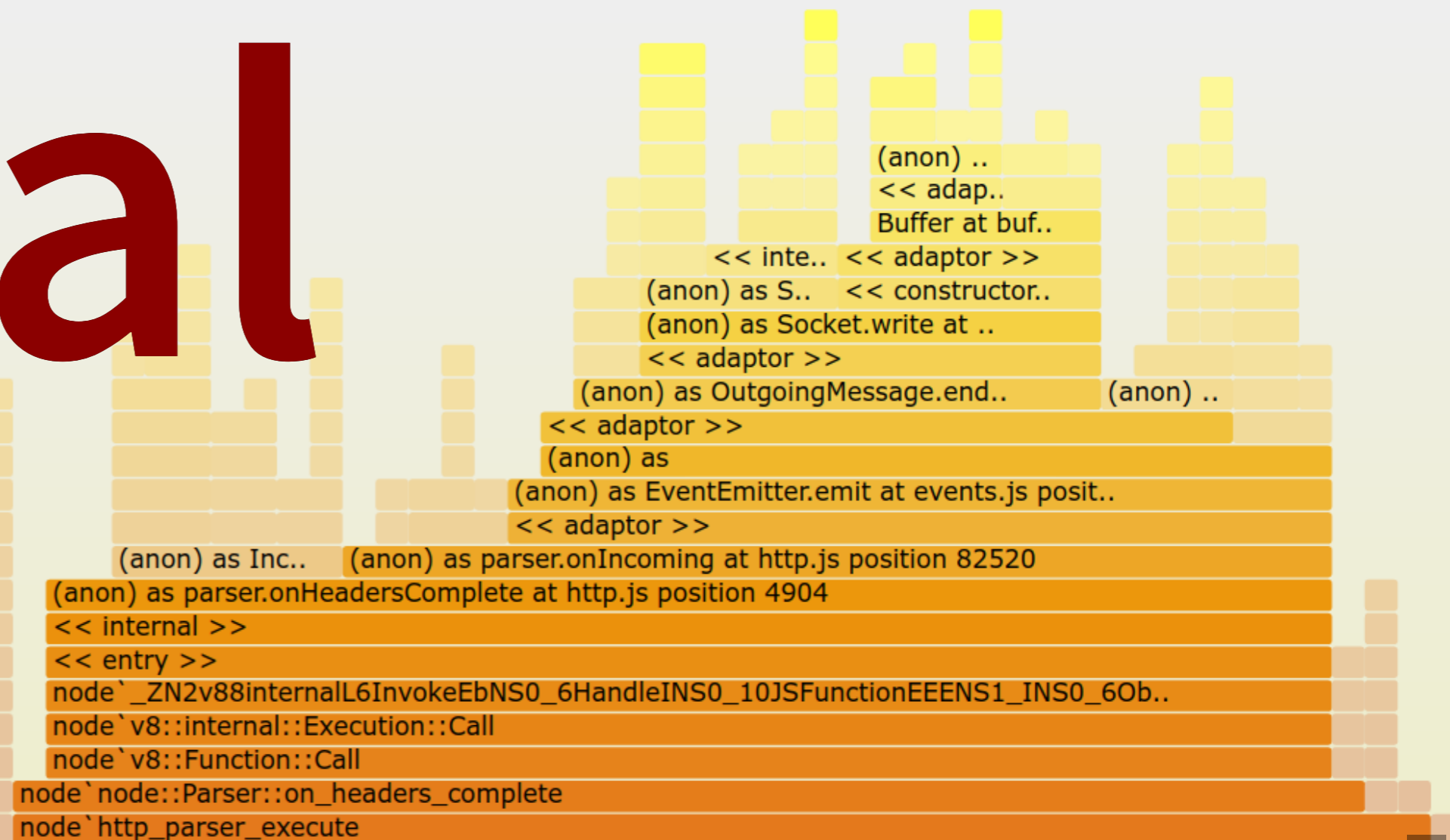
1. Run your Node.js program
2. In another terminal, run

```
$ dtrace -n 'profile-9  
@[jstack(150, 8000
```

This will sample about

ireal

Flame Graph



WE HAVE GRAPHANA & PROMETSEUS

MAY BE THIS IS IT ?

# Performance Timing API

Stability: 1 - E

The Performance  
purpose of the API  
API as implement

```
const { Per
```

```
const obs =
```





# [[ Scope ]]

```
▶ constructor: f Function() ←  
  length: 0  
  name: ""  
▶ toString: f toString()  
▶ Symbol(Symbol.hasInstance): f [Symbol.hasInstance]()  
▶ get arguments: f ()  
▶ set arguments: f ()  
▶ get caller: f ()  
▶ set caller: f ()  
▶ 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

## 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.

# INSTALLING ! prod & expensive



npm install fibers

0 dependencies  
233 dependents

version 3.0.0  
updated 3 months ago

weekly downloads

23,579



version

3.0.0

license

MIT

weekly downloads

23,579

version

3.0.0

open issues

9

# THREADS...

## Worker Threads

Stability: 1 - Experimental

# EXPERIMENTAL

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

Workers are useful for performing CPU-intensive JavaScript operations; do not use them for I/O, since Node.js's built-in performance optimizations are already faster than worker threads can.

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







# CONTROL FLOW

[https://en.wikipedia.org/wiki/Control\\_flow](https://en.wikipedia.org/wiki/Control_flow)

# GREEN THREADS

[https://en.wikipedia.org/wiki/Green\\_threads](https://en.wikipedia.org/wiki/Green_threads)

**RUBY, GO, CLOSURE, LISP ...**

# SMALLTALK

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



# TRACKING

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

Иван Тулуп:  
асинхронщина в JS  
ПОД КАПОТОМ  
Михаил Башуров

**FC**  
2018

**Frontend  
Conf**

Профессиональная  
конференция  
фронтенд-разработчиков

[Habr/YouTube](#)



# EVENT LOOP



# EVENT LOOP : NODE

node

HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | NEWS | FOUNDATION

Docs

ES6 and beyond

v10.13.0 API LTS

v11.2.0 API

Guides

## The Node.js Event Loop, Timers, and `process.nextTick()`

What is the Event Loop?

[Edit on GitHub](#)

M | | Follow | Sign in | Get started

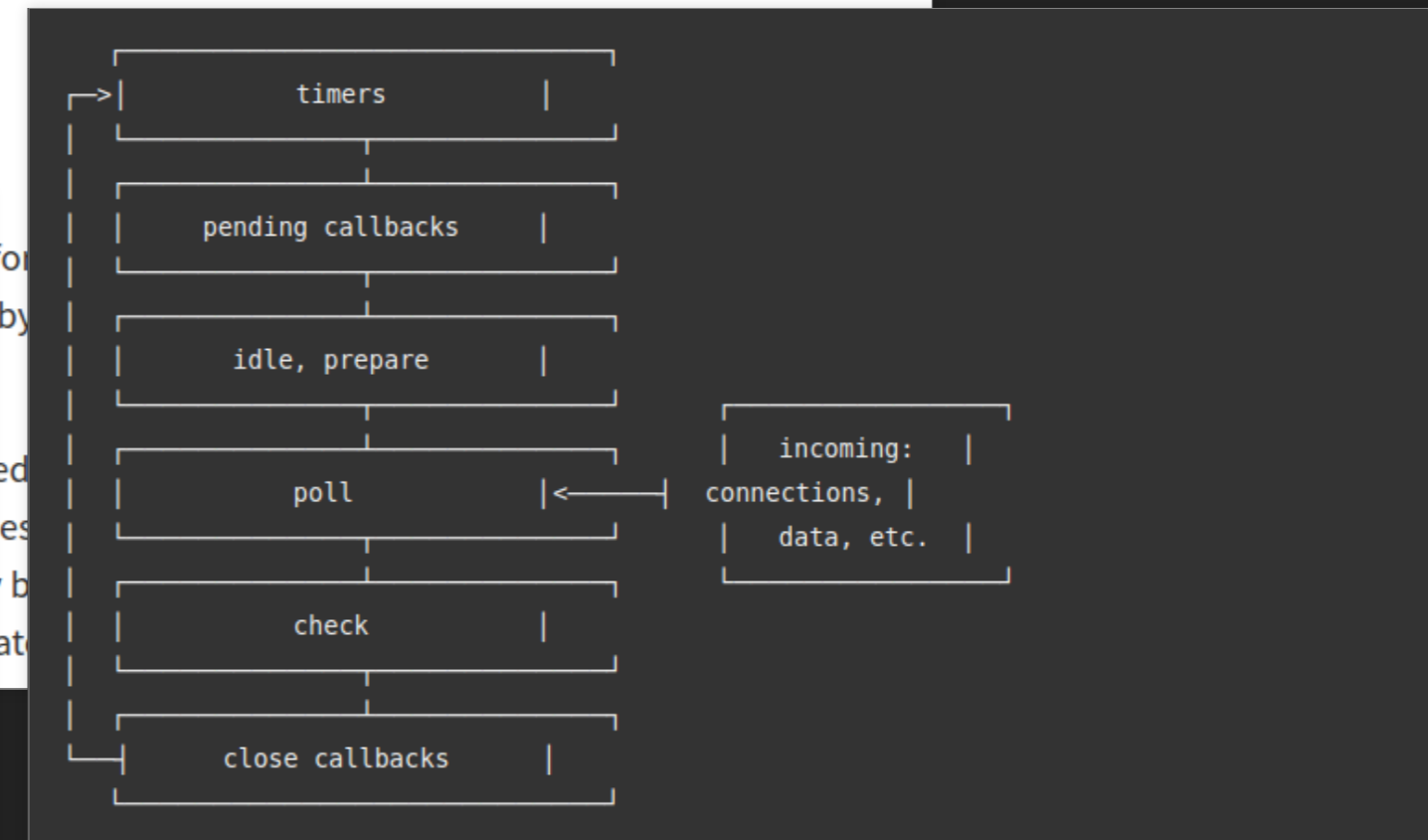
HOME | NODE.JS | ХАРДКОР | ФП | ДЛЯ НАЧИНАЮЩИХ | ПОДКАСТЫ | VUE.JS | УЧАСТИЕ | ПОДДЕРЖИ ПРОЕКТ

## Цикл событий Node.js, таймеры и `process.nextTick()`

Перевод [официальной документации Node.js](#)

### Что такое Event Loop?

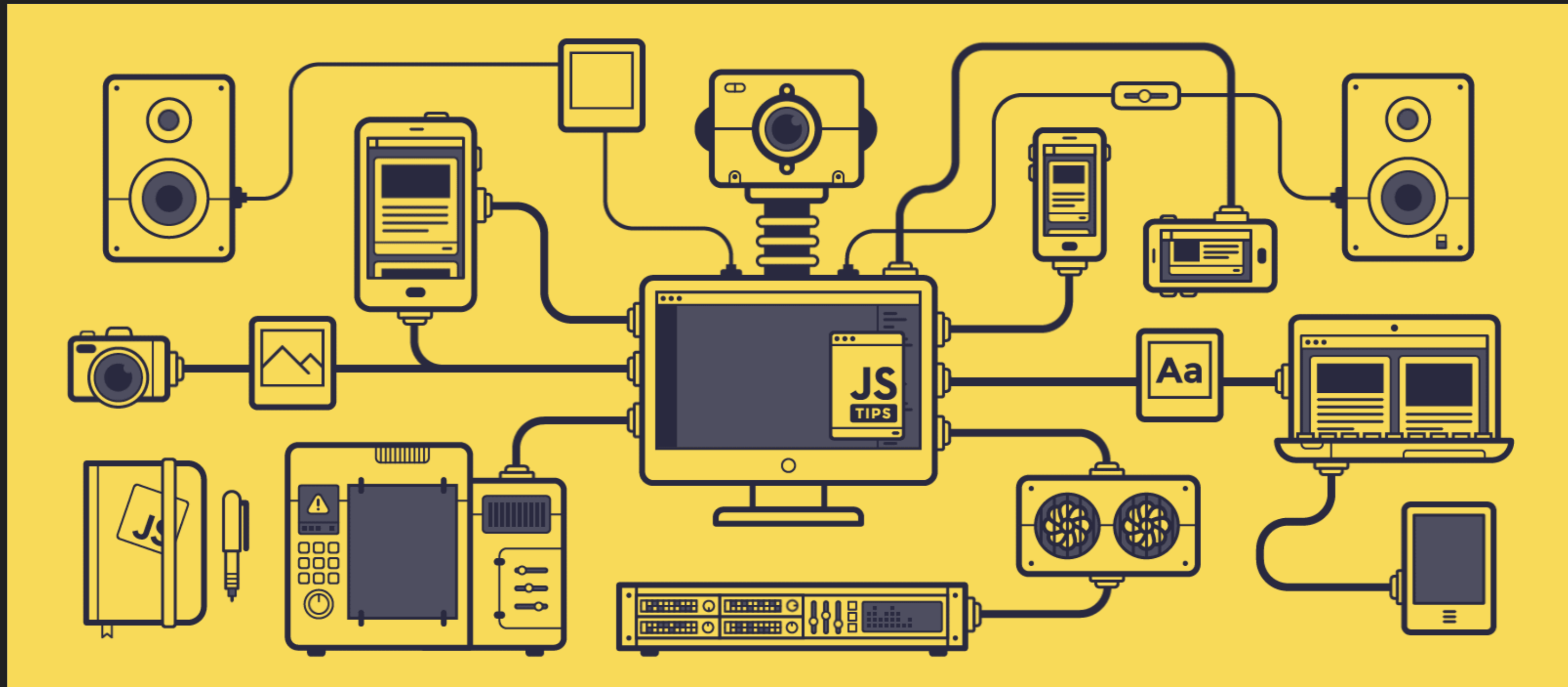
Цикл событий (Event Loop) — это то, что позволяет Node.js выполнять неблокирующие операции ввода/вывода (несмотря на то, что JavaScript является однопоточным) путем выгрузки операций в ядро системы, когда это возможно.



# 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 SEQUENTIAL FLOW**  
**VERY FAST ONE, THOUGH**



WIKIPEDIA  
The Free Encyclopedia

Not logged in [Talk](#) [Contributions](#) [Create account](#) [Log in](#)

[Article](#)

[Talk](#)

[Read](#)

[Edit](#)

[View history](#)

# Von Neumann architecture

From Wikipedia, the free encyclopedia



and Univers  
known as the  
er architectu  
ysicist John  
C.[1] That o  
al computer  
hmetic logi  
ion register  
ons





EVERY ASYNC OPERATION

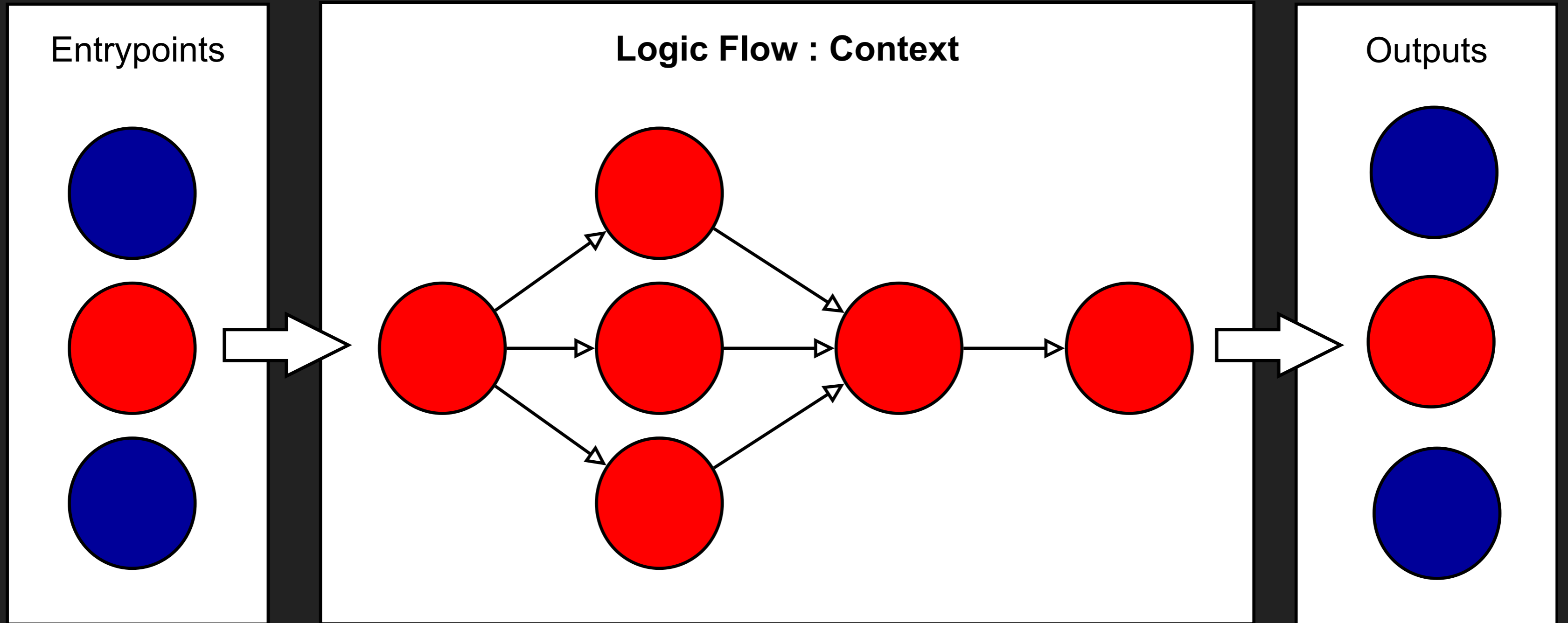
**JUST TRIGGERS**

INTERRUPTION CALLBACK

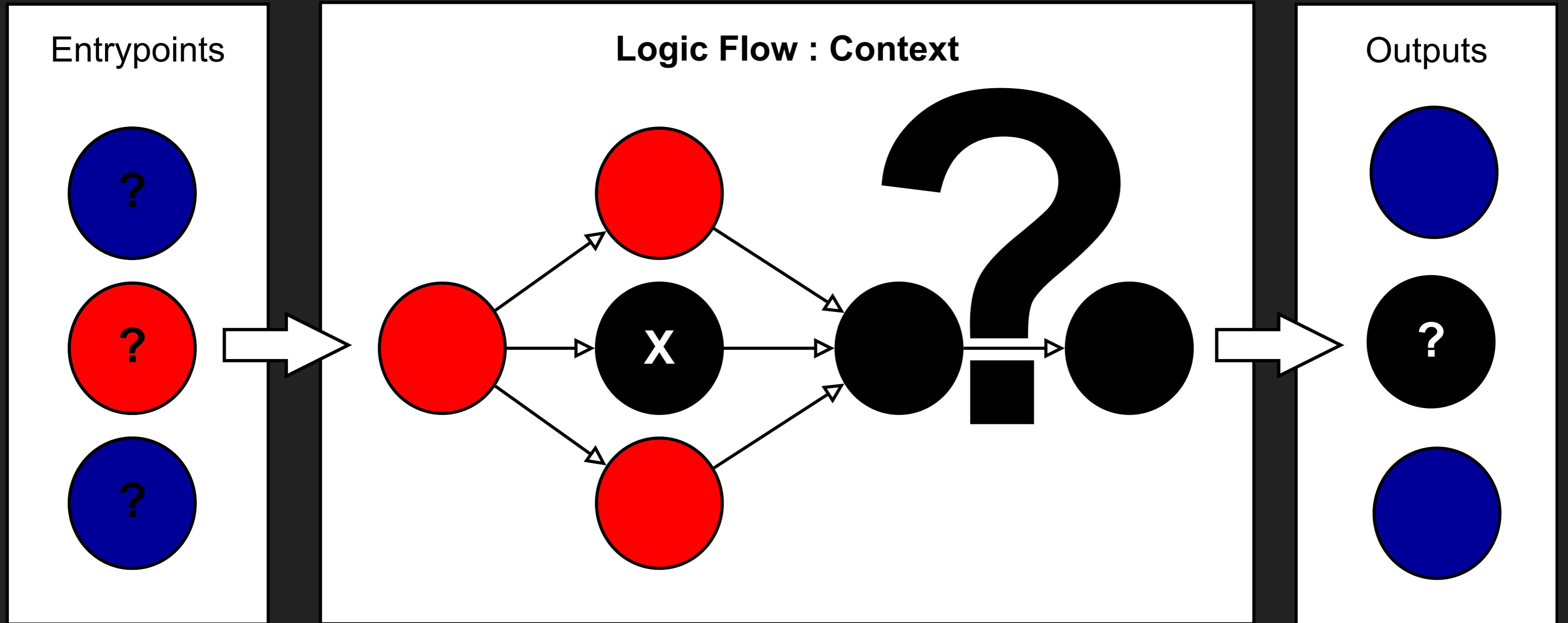
REGISTERED SOMEWHERE

**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



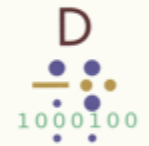

```
process.on('uncaughtException', (err) => {  
  log.error(err);  
  
  // ??? req.res.status(503);  
  
  process.exit(1);  
});
```



# C.O.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 )**

**asynclId**

**triggerId**

**async\_hooks.executionAsynclId()**

**async\_hooks.triggerAsynclId()**

**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, WRITWRAP, 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:

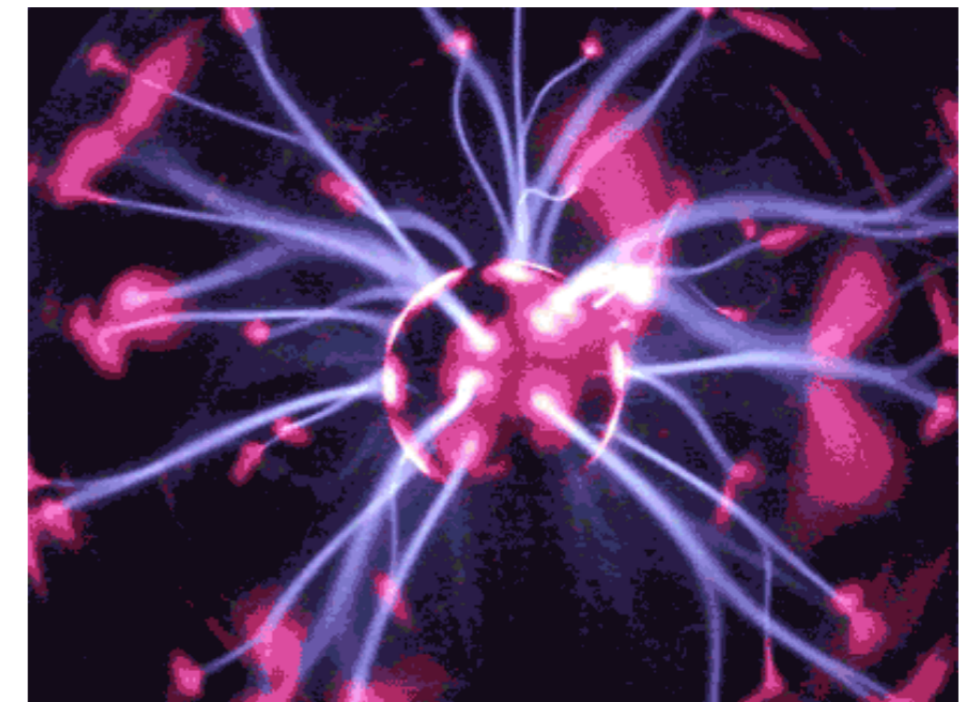
```
init, before, after and  
destroy.
```



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 x
```

0

```
const eid = hooks.eid;
const tid = hooks.tid;
```

```
eid = 1
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

- meaningful 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

The image shows two overlapping screenshots. The background screenshot is the NPM page for the 'zone' package, version 0.3.4, published 4 years ago. It features a 'Readme' button and the title 'StrongLoop zone library'. The foreground screenshot is the GitHub repository page for 'angular / zone.js'. It shows 148 watchers, 2,894 stars, 243 issues, 21 pull requests, and 1 project. The repository description is 'Implements Zones for JavaScript'. It has 730 commits, 8 branches, 94 releases, and 86 contributors. The latest commit is by trevorade and mhevery, titled 'Fix ZoneAwarePromise.all to resolve at the correct time (#1150)'. Below the commit list, there are folders for 'dist' (chore: release v0.8.26) and 'doc' (chore: Fix the URL to zone.js.d.ts (#922)).

**zone**  
0.3.4 • Public • Published 4 years ago

Readme 1 Dependencies 12 Depend

### StrongLoop zone library

### Overview

The Zone library provides a way to represent the dy  
Just like the scope of a function defines where it m  
lifetime that is it active.

angular / zone.js Watch 148 Star 2,894

Code Issues 243 Pull requests 21 Projects 1 Wiki Insights

Implements Zones for JavaScript <https://github.com/angular/zone.js>

730 commits 8 branches 94 releases 86 contributors View

Branch: master New pull request Create new file Upload files Find file Clon

trevorade and mhevery Fix ZoneAwarePromise.all to resolve at the correct time (#1150) Latest commit 9ed5

dist chore: release v0.8.26

doc chore: Fix the URL to zone.js.d.ts (#922)

# GOOGLE CLOUD TRACING CLS

Overview Trace List Analysis Reports

**Insights**  
No insights to report for the past 7 days.

**Most recent traces**

LATENCY	URI	TIME
6 ms	/_monitoring/test...	9:46 AM
5 ms	/_monitoring/test...	9:46 AM
4 ms	/_monitoring/test...	9:46 AM
8 ms	/_monitoring/test...	9:46 AM
4 ms	/_monitoring/test...	9:45 AM
24 ms	/books	9:45 AM

[See more ...](#)

**Most frequent URIs**

LATENCY	URI	REPORT
29 ms	/books?pageToken=123	<a href="#">View Report</a>
3,285,166 ms	/_ah/background	
7 ms	/_monitoring/test...	
29 ms	/_monitoring/test...	
6 ms	/_ah/start	

**Most frequent RPCs**

LATENCY	RPC
0 ms	/logservice.Flush
0 ms	/modules.GetNumInstances

**Daily analysis reports**

[Latency density distribution](#) Cumulative distribution

**Overall latency for requests that make remote procedure calls**  
/books?pageToken=123

% of total requests

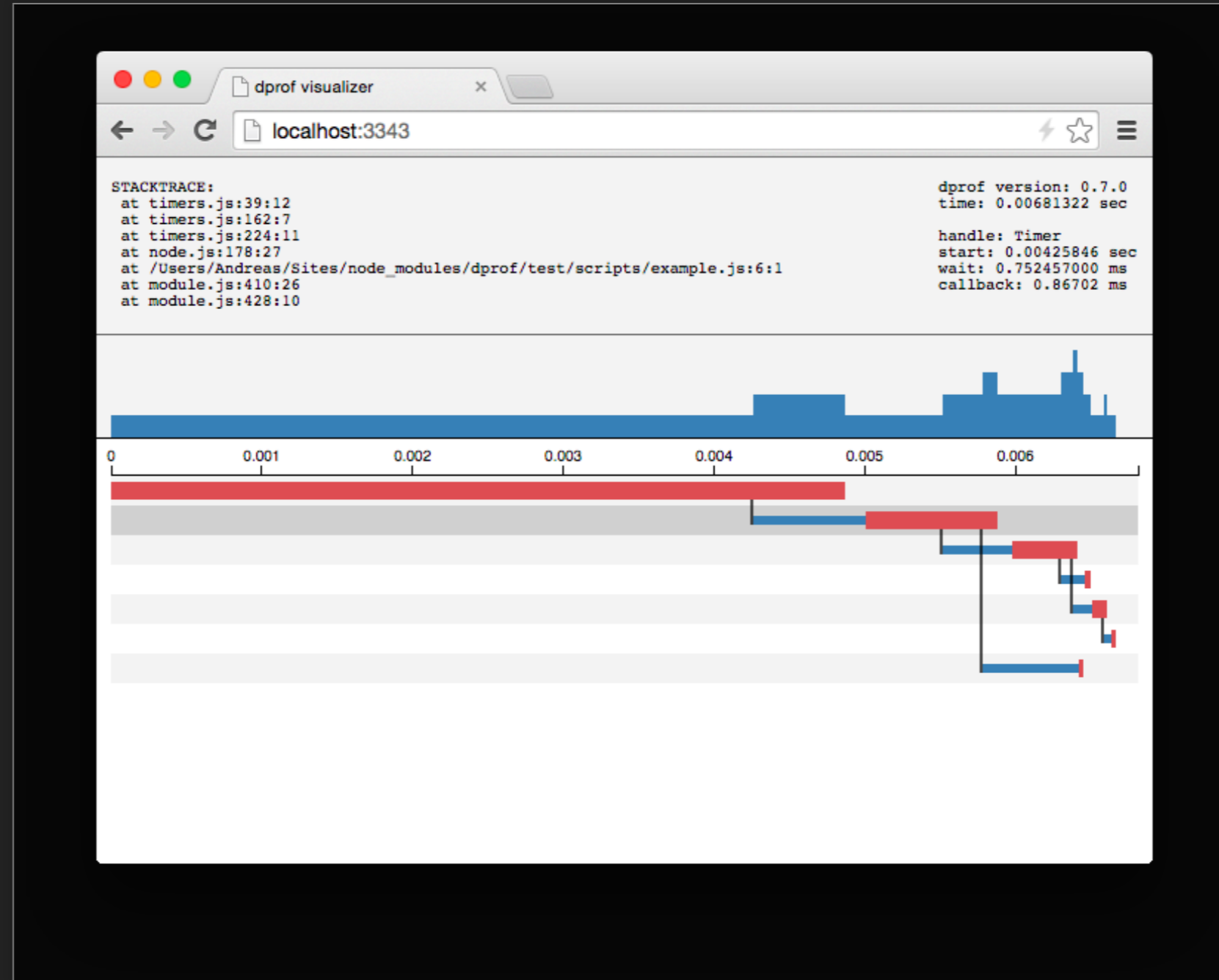
Start: 2016-04-11 (17:00:00) End: 2016-04-12 (17:00:00)  
Module: All Version: All

**Latency**

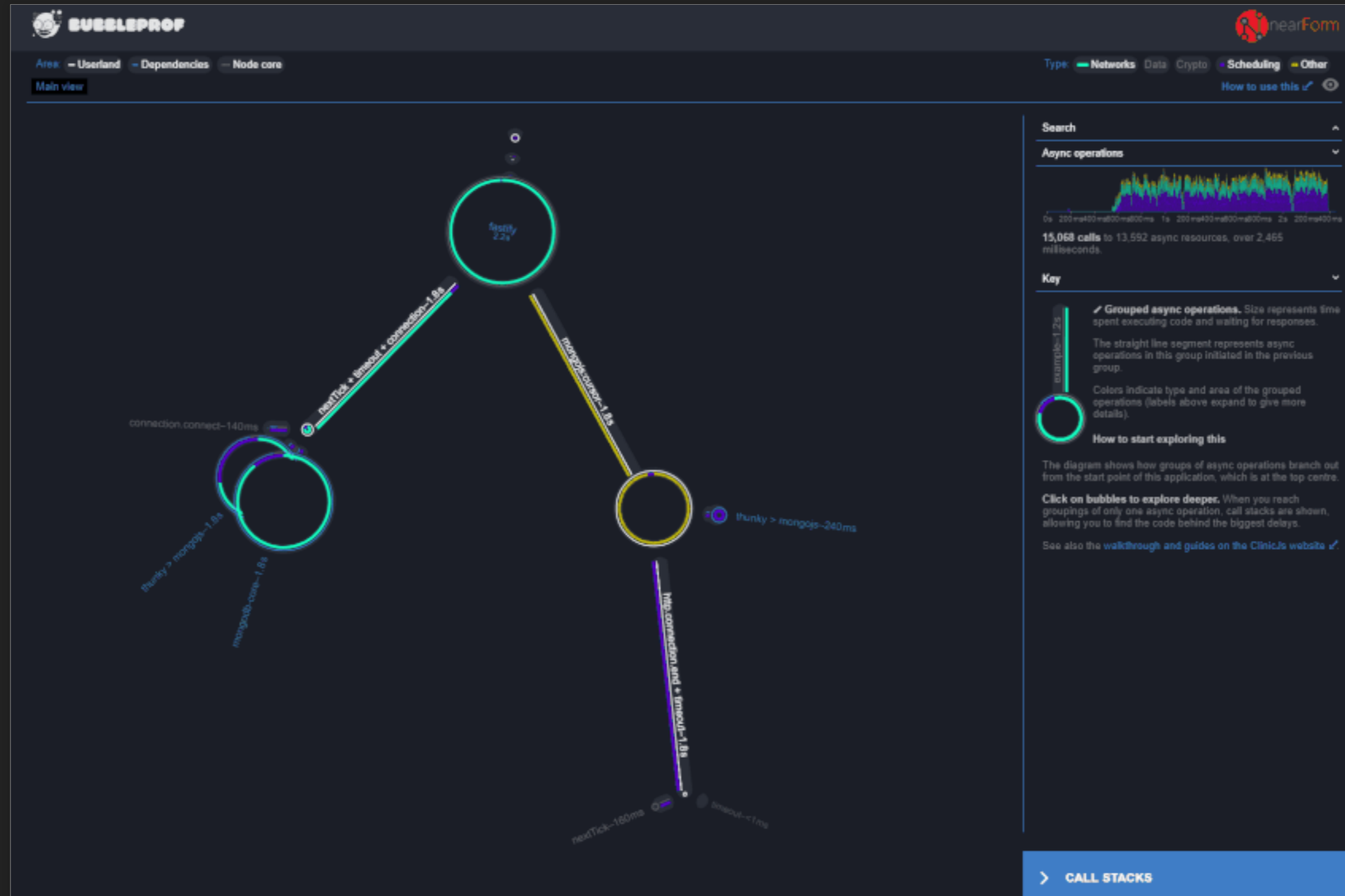
Request percentile	Latency	Sample Traces
25%	15 ms	<a href="#">1</a> <a href="#">2</a> <a href="#">3</a> <a href="#">4</a>
50%	18 ms	<a href="#">1</a> <a href="#">2</a> <a href="#">3</a> <a href="#">4</a>
90%	30 ms	<a href="#">1</a> <a href="#">2</a> <a href="#">3</a> <a href="#">4</a>
95%	38 ms	<a href="#">1</a> <a href="#">2</a> <a href="#">3</a> <a href="#">4</a>
98%	55 ms	<a href="#">1</a> <a href="#">2</a> <a href="#">3</a> <a href="#">4</a>
99%	618 ms	<a href="#">1</a> <a href="#">2</a> <a href="#">3</a> <a href="#">4</a>

[Create analysis report](#)

# ANDREAS MADSEN DPROF



# CLINC + BUBBLEPROF



released

dive to callback with context

— seems just a nuance —

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	test src change a day ago
example	test src change a day ago
src	2.2.0 keys 12 days ago
test	add a day ago
.eslintrc.js	function fix test init file 2 months ago
.gitignore	bug fixing & hopAutoWrap a month ago
README.md	Update README.md 19 days ago
package.json	test src change a day ago

not everything

but something

still wrong



# LIMITATIONS







# HOW IT STARTED : ASYNC-LISTENER



**Trevor Norris**  
The Birth and Evolution of AsyncHooks

**JS UtahJS Conference | 2017**  
#utahjs17

# 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 different 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 hookSetTimeout = 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.writeFileSync(process.stdout.fd, msg); // doc: SYNC !
```

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

```
// Track from Pointer function  
// have to make own wrapper  
// 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](https://en.wikipedia.org/wiki/Escape_analysis)

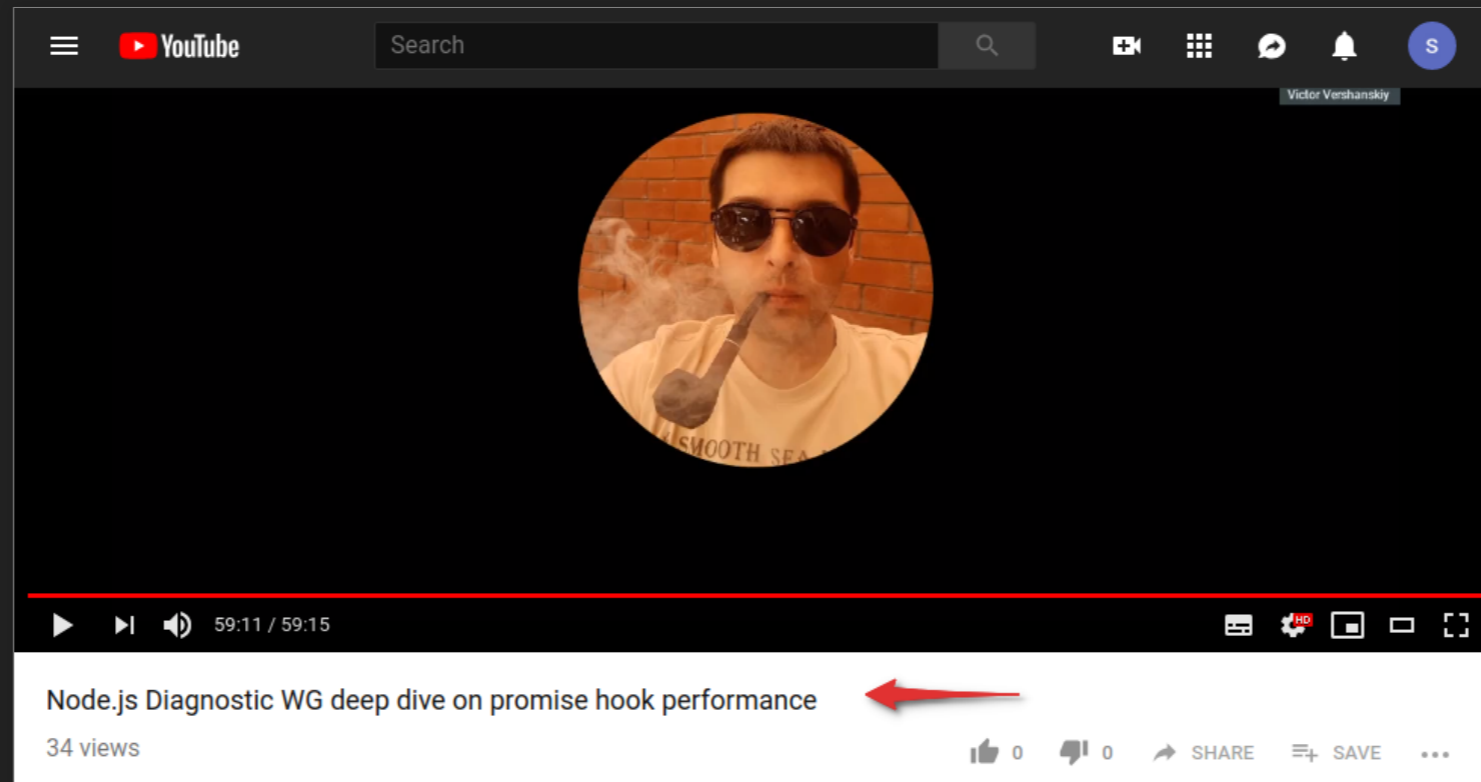
## SHAPE ANALYSIS (PROGRAM ANALYSIS)

[https://en.wikipedia.org/wiki/Shape\\_analysis\\_\(program\\_analysis\)](https://en.wikipedia.org/wiki/Shape_analysis_(program_analysis))

## ALIAS ANALYSIS

[https://en.wikipedia.org/wiki/Alias\\_analysis](https://en.wikipedia.org/wiki/Alias_analysis)

PROMISES : **ISSUE # 248**  
PERFORMANCE IS NOT SO GOOD  
BUT THEY ARE FIXING







# 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

othiym23 / **node-continuation-local-storage**

<> Code Issues 36 Pull requests 8 Projects

The

**Open**

Opened this issue



**Ilya Kantor** iliakan

Opened this issue



**ofrobots** commented on 14 Apr 2016

`AsyncWrap` is indeed a good alternative to `async-listener`, but it doesn't provide enough functionality to cover the continuation-local-storage use cases.

OP mentioned 'the patching problem'. I call it the 'user space queuing problem' ([more details here](#)). It stems from the fact that any user space code (JS or native) could queue callbacks can resume them in a context different. The status quo here has been to discover such code, and get them monkey patch them to ensure they continue working. This is a never ending battle.

IMO a longer term solution would be to



# 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







# CHROME DEBUG PROTOCOL : EXAMPLES

Version  
v8-inspector (node)

## Chrome DevTools Protocol Viewer

Domains

Console

Debugger

HeapProfiler

Profiler

Runtime


Schema

### Runtime Domain

Runtime domain exposes JavaScript runtime by means of `Runtime` domain. `Runtime` domain can be used for further object reference. Original object reference can be used for further object reference.

### Methods

`Runtime.awaitPromise`



# NO WAY **EVENT TRACING** INTERCEPTION LIVE DEMOS!

-- trace flag : test.js

%DebugTrace(); : testDT.js

# ASKING ANY HELP

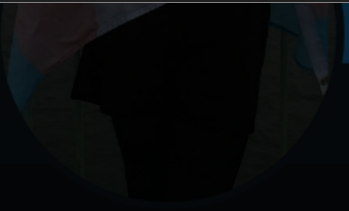
Anna Henningsen  
Andreas Madsen  
Tomas Watson

Async Hooks CLS review request context-dive x

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

**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

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...](https://nodejs.org/api/async_hooks...)  
Oct 20

Kind Regards  
Really Sorry for interrupting

Hi!  
Extremely great you answe

question about async\_hooks

**system**  
Hi Tomas! Sorry for interrupting! 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

The screenshot shows a web browser window with the URL <https://cloud.google.com/trace/docs/quickstart>. The page header includes the Google Cloud logo and navigation links: Why Google, Products, Solutions, Pricing, Security, Documentation (highlighted), Customers, and Partne. There are also buttons for 'Contact sales' and 'Try free'. The main content area features a breadcrumb trail: Stackdriver > Documentation > Trace > Documentation. The title is 'Quickstart' with a 5-star rating and a 'SEND FEEDBACK' link. The introductory text states: 'This page shows you how to perform basic operations for a project that has already been instrumented with Stackdriver Trace.' The main body text explains: '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.' A 'Before you begin' section starts with the step: '1. Select or create a GCP project.' A blue button at the bottom says 'GO TO THE MANAGE RESOURCES PAGE'. The left sidebar contains a 'Stackdriver Trace' menu with 'Quickstart' selected, and a 'How-to Guides' section with 'Setting Up Stackdriver Trace' and 'Viewing Traces' expanded. The right sidebar has a 'Contents' table of contents.

Quickstart | Stackdriver Trac x +

← → ↻ <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 > Documentation > Trace > Documentation

☆☆☆☆☆ [SEND FEEDBACK](#)

## Quickstart

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.

[GO TO THE MANAGE RESOURCES PAGE](#)

**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**

**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>

```
110     patch: function(mongo, api) {
111         shimmer.wrap(mongo.Server.prototype, 'command', wrapWithLabel(api, 'mongo-command'));
112         shimmer.wrap(mongo.Server.prototype, 'insert', wrapWithLabel(api, 'mongo-insert'));
113         shimmer.wrap(mongo.Server.prototype, 'update', wrapWithLabel(api, 'mongo-update'));
114         shimmer.wrap(mongo.Server.prototype, 'remove', wrapWithLabel(api, 'mongo-remove'));
115         shimmer.wrap(mongo.Cursor.prototype, 'next', createNextWrap(api));
116     },
117     unpatch: function(mongo) {
118         shimmer.unwrap(mongo.Server.prototype, 'command');
119         shimmer.unwrap(mongo.Server.prototype, 'insert');
120         shimmer.unwrap(mongo.Server.prototype, 'update');
121         shimmer.unwrap(mongo.Server.prototype, 'remove');
122         shimmer.unwrap(mongo.Cursor.prototype, 'next');
123     }
```

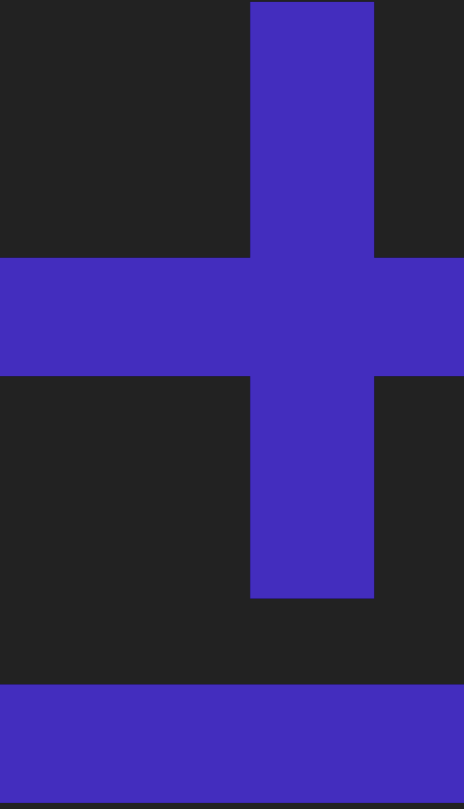
# LIVE DEMOS!

mongoose polling jump

async code branch jump



**RESUME**





## PROS

- LTS works
- tracking
- `async ()`

## CONS

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



# CLS NPM

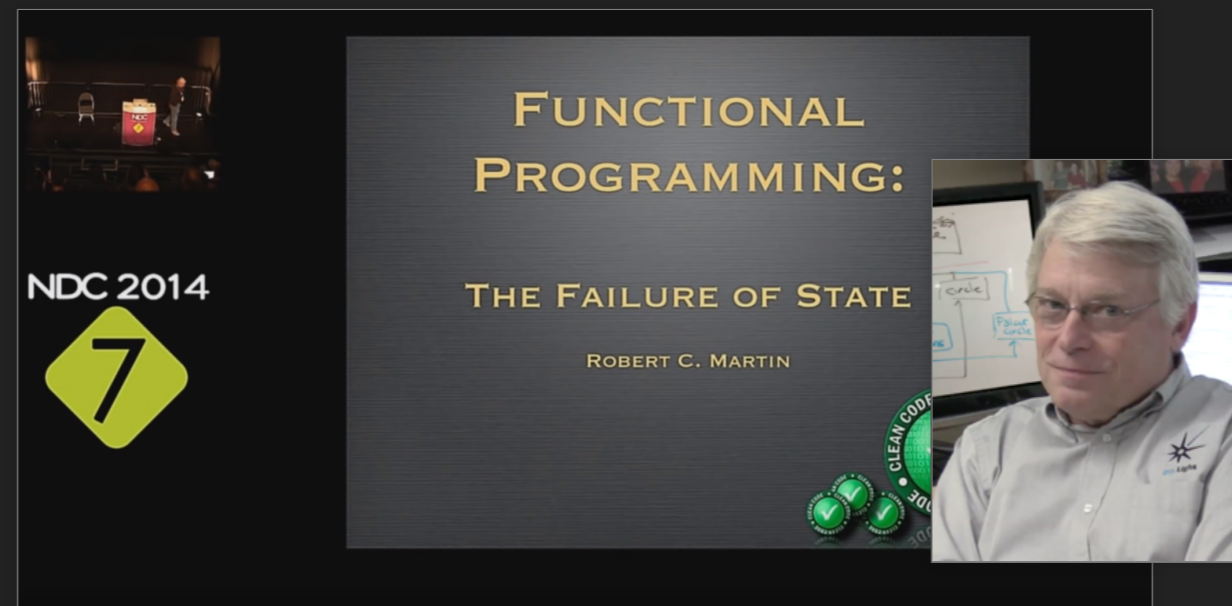
cls-hooked

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`



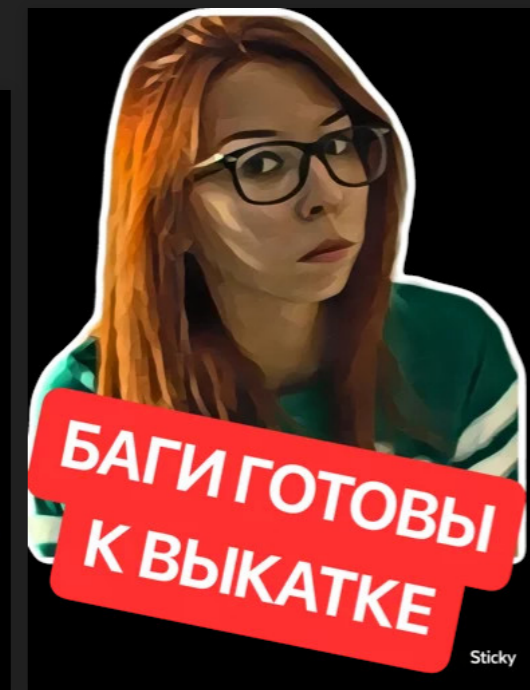
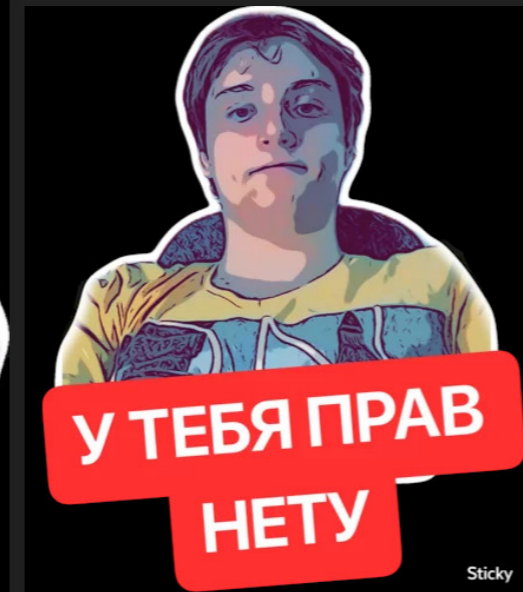
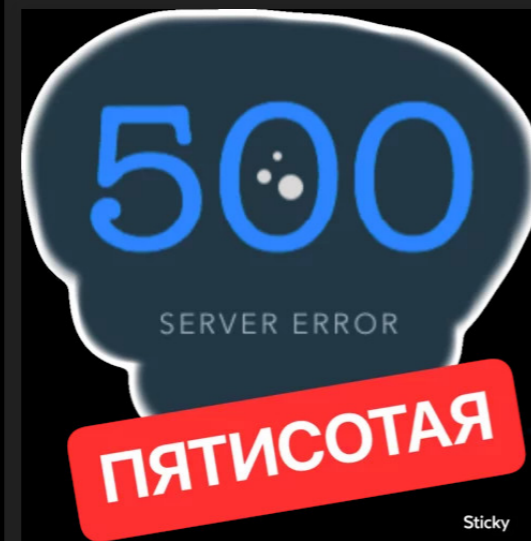
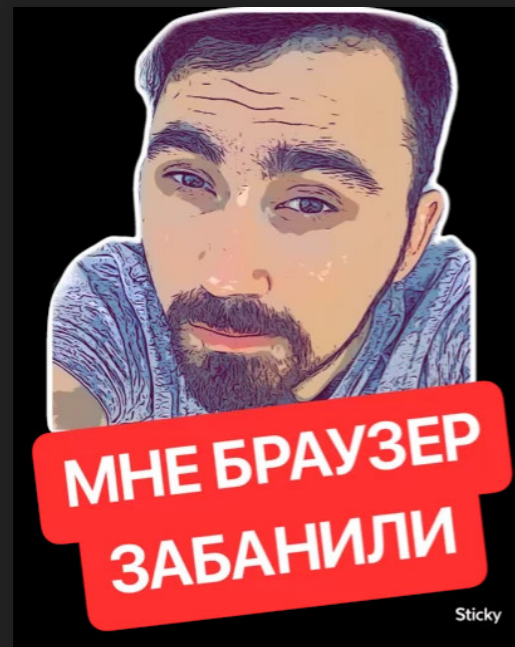
## AND SPECIAL THANKS TO

- [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 & examptes
- [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!