Thomas Watson

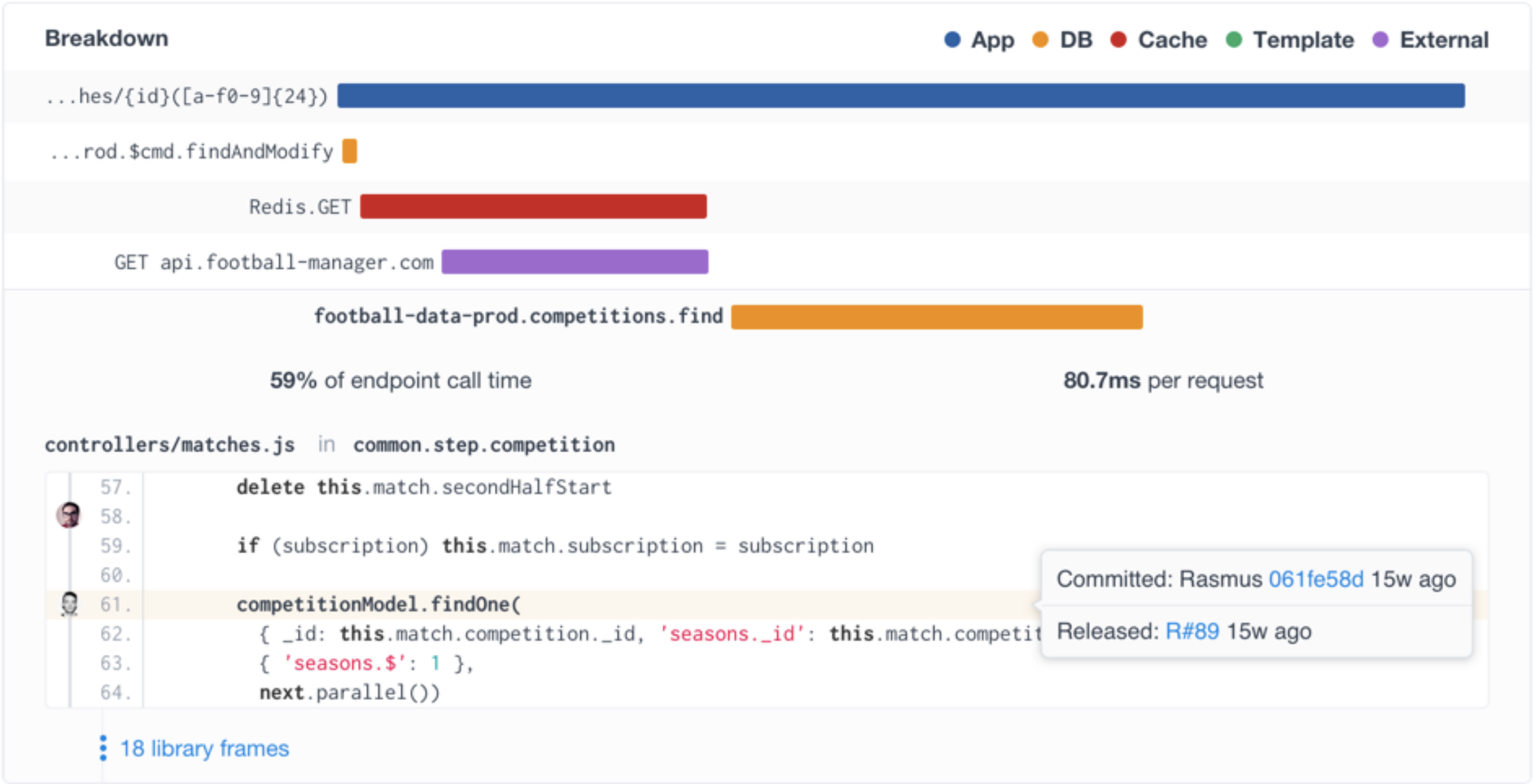
@wa7son

github.com/watson

@wa7son

Detailed activity breakdown

Simple performance breakdown that shows you where your app needs optimizing, like SQL queries, MongoDB queries, http requests to other services, etc.



@wa7son

@wa7son

@wa7son

NASA

Pixar

@wa7son

github.com/watson/talks

@wa7son

Instrumenting Node.js in production

@wa7son

Incoming Request

Response

Transaction

HTTP Request lifetime

BOOM!

Traces

@wa7son

Normal solutions

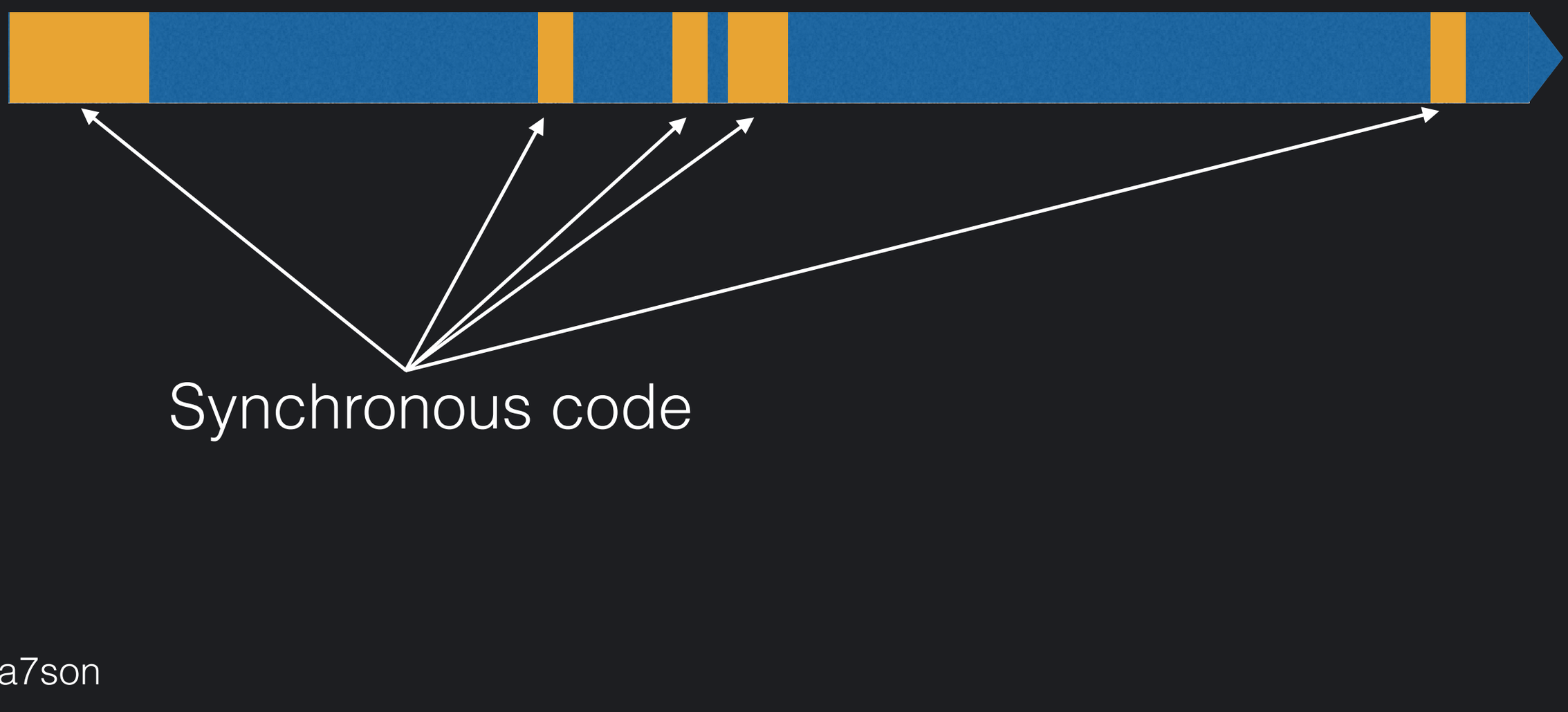
- ~~Use a global or singleton object to store context~~
- ~~Pass a context object around between function calls~~
- ???

The Event Loop

@wa7son

nodejsreactions.tumblr.com

The Event Loop



The Event Loop

@wa7son

Goals

- Track HTTP request as transactions
- Instrument I/O
- Instrument potential CPU expensive operations
- Associate exceptions with transactions
- Don’t manually pass context around in the app
- Plug’n’play
- Almost no overhead

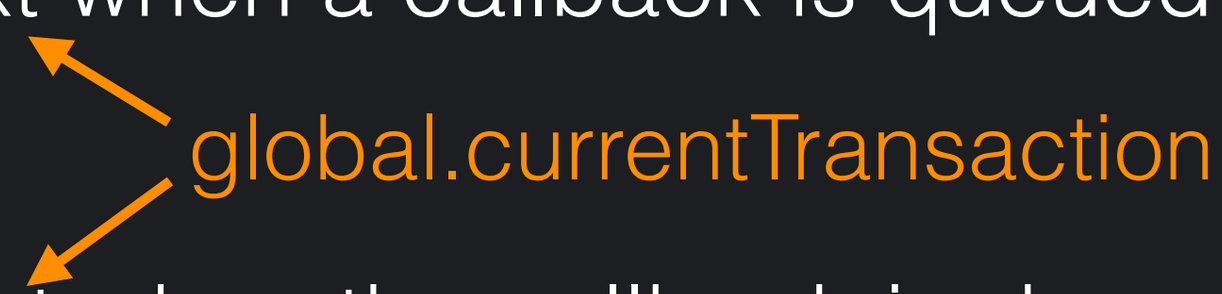
Problems

- Multiple HTTP requests active at the same time
- Not possible to associate exceptions with origin HTTP request
- No API to pass context across the async boundary

Solution

global.currentTransaction = new Transaction(req)

1. Record context when a callback is queued on the event loop
2. Restore context when the callback is de-queued from the event loop



@wa7son

Patch the core

- Patch Module._load
- Patch the HTTP server to access new requests
- Patch **every** async operation
  - timers
  - process.nextTick
  - Promise (native)
  - libuv

github.com / watson / require-in-the-middle

@wa7son

AsyncWrap

https://github.com/nodejs/tracing-wg

```
var asyncWrap = process.binding('async_wrap')
```

```
var asyncWrap = process.binding('async_wrap')

asyncWrap.setupHooks({init, pre, post, destroy})
asyncWrap.enable()
```

```
var asyncWrap = process.binding('async_wrap')

asyncWrap.setupHooks({init, pre, post, destroy})
asyncWrap.enable()

function init (uid, provider, parentUid, parentHandle) {
  log('async_wrap: init')
}
function pre (uid) {
  log('async_wrap: pre')
}
function post (uid) {
  log('async_wrap: post')
}
function destroy (uid) {
  log('async_wrap: destroy')
}
```

@wa7son

```
function post (uid) {
  log('async_wrap: post')
}
function destroy (uid) {
  log('async_wrap: destroy')
}

var fs = require('fs')

log('user: before')
fs.open(__filename, 'r', function (err, fd) {
  log('user: done')
})
log('user: after')
```

```
var asyncWrap = process.binding('async_wrap')

asyncWrap.setupHooks({init, pre, post, destroy})
asyncWrap.enable()

function init (uid, provider, parentUid, parentHandle) {
  log('async_wrap: init')
}
function pre (uid) {
  log('async_wrap: pre')
}
function post (uid) {
  log('async_wrap: post')
}
function destroy (uid) {
  log('async_wrap: destroy')
}

var fs = require('fs')

log('user: before')
fs.open(__filename, 'r', function (err, fd) {
  log('user: done')
})
log('user: after')
```

@wa7son

```
var asyncWrap = process.binding('async_wrap')

asyncWrap.setupHooks({init, pre, post, destroy})
asyncWrap.enable()

function init (uid, provider, parentUid, parentHandle) {
  log('async_wrap: init')
}
function pre (uid) {
  log('async_wrap: pre')
}
function post (uid) {
  log('async_wrap: post')
}
function destroy (uid) {
  log('async_wrap: destroy')
}

var fs = require('fs')

log('user: before')
fs.open(__filename, 'r', function (err, fd) {
  log('user: done')
})
log('user: after')
```

```
user: before
async_wrap: init
user: after
async_wrap: pre
user: done
async_wrap: post
async_wrap: destroy
```

@wa7son

```
var asyncWrap = process.binding('async_wrap')

asyncWrap.setupHooks({init, pre, post, destroy})
asyncWrap.enable()

function init (uid, provider, parentUid, parentHandle) {
  console.log('async_wrap: init')
}
function pre (uid) {
  console.log('async_wrap: pre')
}
function post (uid) {
  console.log('async_wrap: post')
}
function destroy (uid) {
  console.log('async_wrap: destroy')
}

var fs = require('fs')

console.log('user: before')
fs.open(__filename, 'r', function (err, fd) {
  console.log('user: done')
})
console.log('user: after')
```

@wa7son

```
var asyncWrap = process.binding('async_wrap')

asyncWrap.setupHooks({init, pre, post, destroy})
asyncWrap.enable()

function init (uid, provider, parentUid, parentHandle) {
  console.log('async_wrap: init')
}
function pre (uid) {
  console.log('async_wrap: pre')
}
fun
  c
}
function destroy (uid) {
  console.log('async_wrap: destroy')
}

var fs = require('fs')

console.log('user: before')
fs.open(__filename, 'r', function (err, fd) {
  console.log('user: done')
})
console.log('user: after')
```

```
FATAL ERROR: node::AsyncWrap::AsyncWrap init hook threw
```

@wa7son

```
fs.writeSync(1, util.format('%s\n', msg))
```

@wa7son

```
fs.writeSync(1, util.format('%s\n', msg))
```

```
process._rawDebug(msg)
```

```
var asyncWrap = process.binding('async_wrap')

asyncWrap.setupHooks({init, pre, post, destroy})
asyncWrap.enable()

function init (uid, provider, parentUid, parentHandle) {
  process._rawDebug('async_wrap: init')
}
function pre (uid) {
  process._rawDebug('async_wrap: pre')
}
function post (uid) {
  process._rawDebug('async_wrap: post')
}
function destroy (uid) {
  process._rawDebug('async_wrap: destroy')
}

var fs = require('fs')

process._rawDebug('user: before')
fs.open(__filename, 'r', function (err, fd) {
  process._rawDebug('user: done')
})
process._rawDebug('user: after')
```

```
user: before
async_wrap: init
user: after
async_wrap: pre
user: done
async_wrap: post
async_wrap: destroy
```

@wa7son

```
function init (uid, provider, parentUid, parentHandle) {
  // this         => current handle
  // uid          => 1, 2, 3...
  // provider     => 0 - 23 (asyncWrap.Providers)
  // parentUid    => 1, 2, 3...
  // parentHandle => parent `this`
}
```

@wa7son

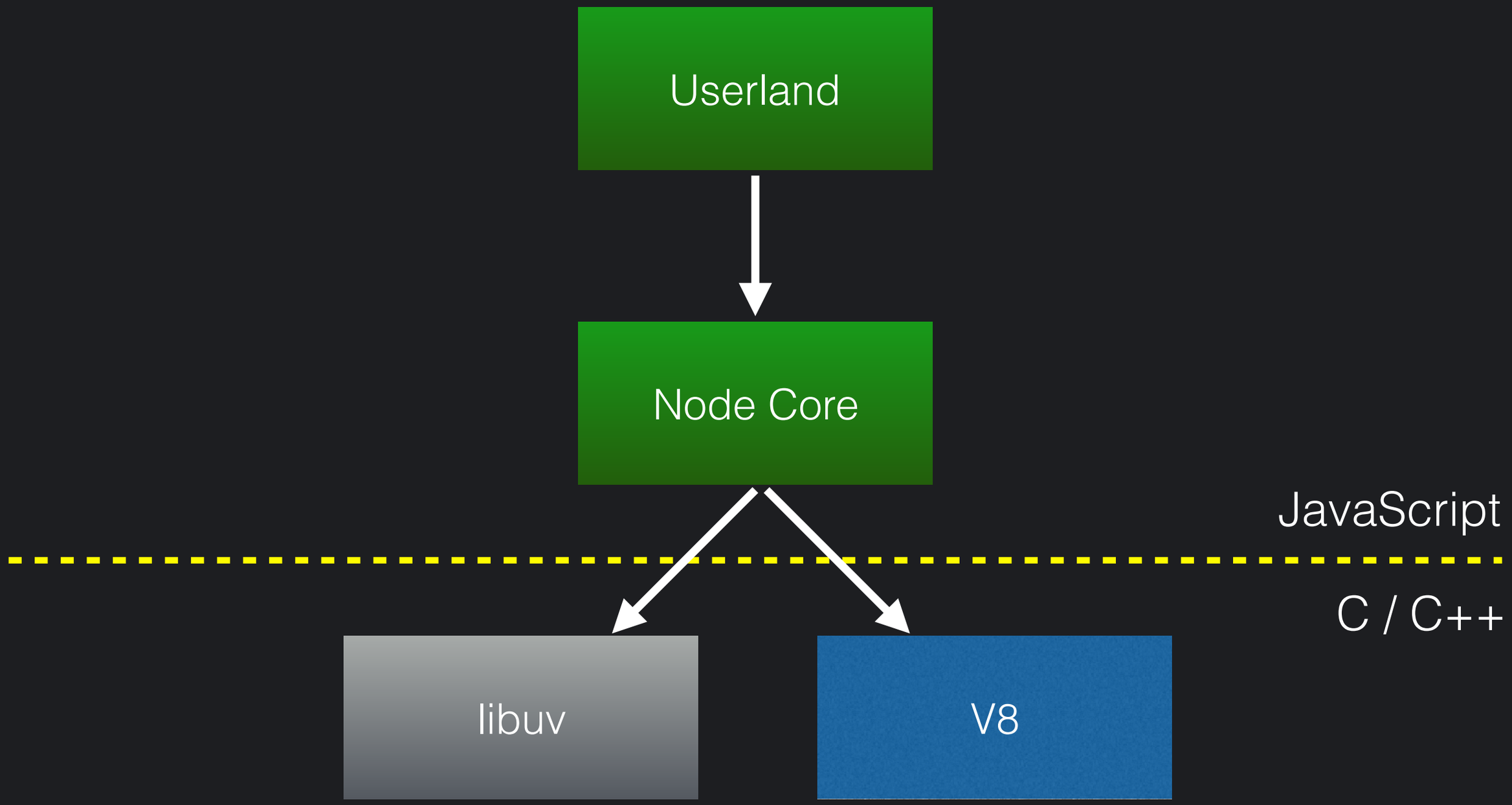
```
> var asyncWrap = process.binding('async_wrap')
undefined
> asyncWrap.Providers
{ NONE: 0,
  CRYPTO: 1,
  FSEVENTWRAP: 2,
  FSREQWRAP: 3,
  GETADDRINFOREQWRAP: 4,
  GETNAMEINFOREQWRAP: 5,
  HTTPPARSER: 6,
  JSSTREAM: 7,
  PIPEWRAP: 8,
  PIPECONNECTWRAP: 9,
  PROCESSWRAP: 10,
  QUERYWRAP: 11,
  SHUTDOWNWRAP: 12,
  SIGNALWRAP: 13,
  STATWATCHER: 14,
  TCPWRAP: 15,
  TCPCONNECTWRAP: 16,
  TIMERWRAP: 17,
  TLSWRAP: 18,
  TTYWRAP: 19,
  UDPWRAP: 20,
  UDPSENDWRAP: 21,
  WRITEWRAP: 22,
  ZLIB: 23 }
```

@wa7son

```
function init (uid, provider, parentUid, parentHandle) {
  // this         => current handle
  // uid          => 1, 2, 3...
  // provider     => 0 - 23 (asyncWrap.Providers)
  // parentUid    => 1, 2, 3...
  // parentHandle => parent `this`
}
```

@wa7son

Handle Objects



Handle Objects

```
const TCPConnectWrap = process.binding('tcp_wrap').TCPConnectWrap;
const TCP = process.binding('tcp_wrap').TCP;

const req = new TCPConnectWrap();
req.oncomplete = oncomplete;
req.address = address;
req.port = port;

const socket = new TCP();
socket.onread = onread;
socket.connect(req, address, port);

// later
socket.destroy();
```

Handle Objects

```
const TCPConnectWrap = process.binding('tcp_wrap').TCPConnectWrap;
const TCP = process.binding('tcp_wrap').TCP;

const req = new TCPConnectWrap();
req.oncomplete = oncomplete;
req.address = address;
req.port = port;

const socket = new TCP();
socket.onread = onread;
socket.connect(req, address, port);

// later
socket.destroy();
```

req === this

socket === this

Timers

```
log('user: before #1')
setTimeout(function () {
  log('user: done #1')
}, 2000)
log('user: after #1')

log('user: before #2')
setTimeout(function () {
  log('user: done #2')
}, 2000)
log('user: after #2')
```

Timers

```
log('user: before #1')
setTimeout(function () {
  log('user: done #1')
}, 2000)
log('user: after #1')

log('user: before #2')
setTimeout(function () {
  log('user: done #2')
}, 2000)
log('user: after #2')
```

```
user: before #1
async_wrap: init
user: after #1
user: before #2
user: after #2
```

Timers

```
log('user: before #1')
setTimeout(function () {
  log('user: done #1')
}, 2000)
log('user: after #1')

log('user: before #2')
setTimeout(function () {
  log('user: done #2')
}, 2000)
log('user: after #2')
```

```
user: before #1
async_wrap: init
user: after #1
user: before #2
user: after #2

async_wrap: pre
user: done #1
user: done #2
async_wrap: post
async_wrap: destroy
```

@wa7son


@wa7son

Show some real code

Sh[illegible]ode

github.com/watson/talks

@wa7son

AsyncWrap Gotchas

- Handle creation time
- console.log
- process.nextTick
- Timers
- Promises
- Multiple AsyncWrap’s

Callback queues in user-land

@wa7son

ES6 Modules

@wa7son

GitHub, Inc. github.com/nodejs/tracing-wg

This repository Search Pull requests Issues Gist

nodejs / tracing-wg

Unwatch 60 Unstar 75 Fork 14

Code Issues 18 Pull requests 2 Wiki Pulse Graphs

Tracing Working Group

47 commits 1 branch 0 releases 12 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

watson committed with AndreasMadsen Update asyncWrap.Providers list (#55) ... Latest commit d3a41e9 a day ago

docs	Update asyncWrap.Providers list (#55)	a day ago
wg-meetings	updated YouTube link to video owned by node.js	2 days ago
README.md	jeffo request to join	6 months ago

README.md

tracing-wg

Tracing Working Group

Members

@wa7son

https://github.com/nodejs/tracing-wg

GitHub, Inc. github.com/nodejs/tracing-wg/tree/master/

This repository Search Pull requests Issues Gist

nodejs / tracing-wg

Unwatch 60 Unstar 75 Fork 14

Code Issues 18 Pull requests 2 Wiki Pulse Graphs

Branch: master tracing-wg / docs / AsyncWrap / Create new file Upload files Find file History

watson committed with AndreasMadsen Update asyncWrap.Providers list (#55) ... Latest commit d3a41e9 a day ago

..

example-trace	docs: update after nodejs/node#5756	a month ago
README.md	Update asyncWrap.Providers list (#55)	a day ago

README.md

Node.js tracing - AsyncWrap

AsyncWrap is two things. One is a class abstraction that provides an internal mechanism for handling asynchronous tasks, such as calling a callback. The other part is an API for setting up hooks and allows one to get structural tracing information about the life of handle objects. In the context of tracing the latter is usually what is meant.

The reasoning for the current naming confusion is that the API part implements the hooks through the AsyncWrap class, but this is not inherently necessary. For example if v8 provided those facilities the AsyncWrap class would not need be involved in the AsyncWrap API.

For the remaining description the API part is what is meant by AsyncWrap.

@wa7son https://github.com/nodejs/tracing-wg/tree/master/docs/AsyncWrap

GitHub, Inc. github.com/nodejs/tracing-wg/issues/29

This repository Search Pull requests Issues Gist

nodejs / tracing-wg

Unwatch 60 Unstar 75 Fork 14

Code Issues 18 Pull requests 2 Wiki Pulse Graphs

AsyncWrap issues - overview #29

Edit New issue

Open **AndreasMadsen** opened this issue on Oct 7, 2015 · 8 comments

AndreasMadsen commented on Oct 7, 2015 • edited — Node.js Foundation member

Missing Handle context

- [x] TCPwrap created from server (issue: nodejs/node#2986) - solved by nodejs/node#3216
- [x] HTTP sockets `parserOnBody`
  - issues: nodejs/node#3241, nodejs/node#4416
  - PR: nodejs/node#5419 and nodejs/node#5591, ~~depends on: nodejs/node#4507~~
- [ ] Promises or Microtask in general
  - node issue: nodejs/promises#9
  - v8 issue: https://bugs.chromium.org/p/v8/issues/detail?id=4643
- [ ] nextTick (issue: nodejs/node#666) - should get a new issue
- [ ] setTimeout, setInterval, setImmediate (issue: nodejs/node#666) - should get a new issue
- [ ] addon modules integration with AsyncWrap (~~PR: nodejs/node#3504~~) - needs further discussion.

More events

- [ ] onready (issue: #11) - unlikely to be solved
- [ ] onerror (issue: nodejs/node#669, #7) - awaiting use cases
- [x] ondestructor (PR: nodejs/node#3461)

Labels — None yet

Milestone — No milestone

Assignees — No one—assign yourself

Notifications — Unsubscribe

You're receiving notifications because you're subscribed to this repository.

4 participants

Lock conversation

@wa7son https://github.com/nodejs/tracing-wg/issues/29

Slides & Code

github.com/watson/talks

@wa7son

Mulțumiri!

@wa7son

github.com/watson

opbeat