

Thomas Watson

@wa7son

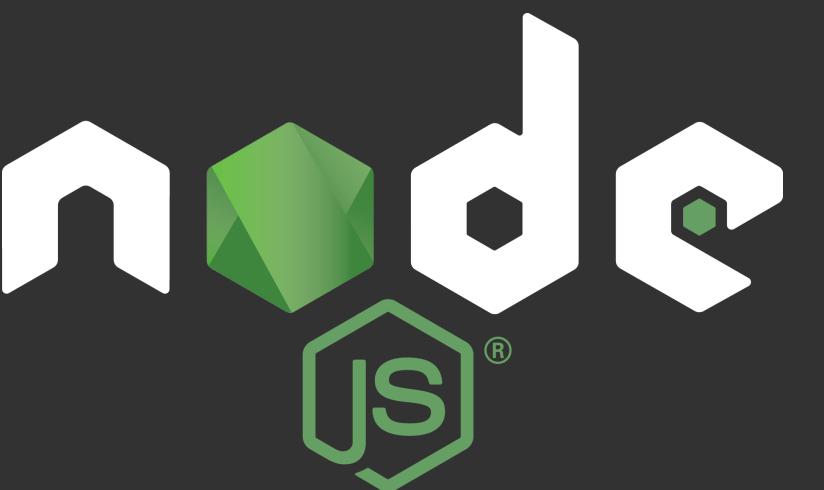
github.com/watson



The Trouble with Tracers

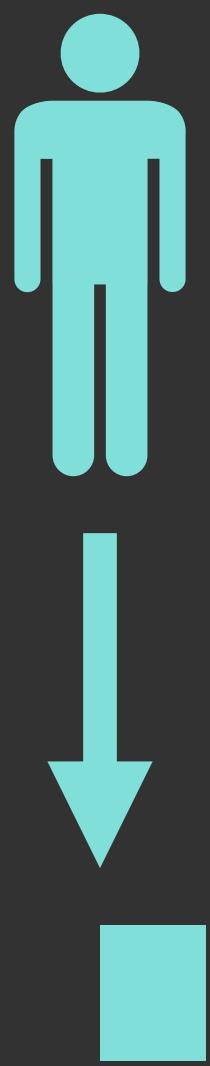
Who am I?

- Thomas Watson
- Open Source developer at github.com/watson
- Principal Software Engineer at Elastic
- Node.js Core Member
- Tweets as @wa7son
- Slides: github.com/watson/talks

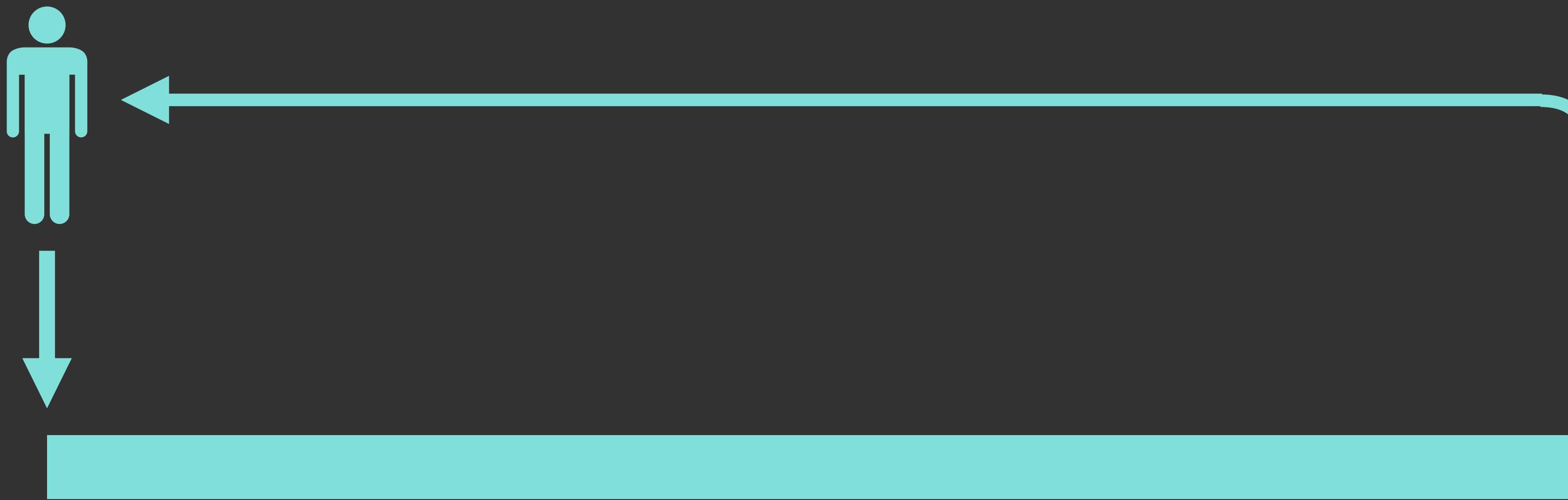


Tracing 101

Tracing 101

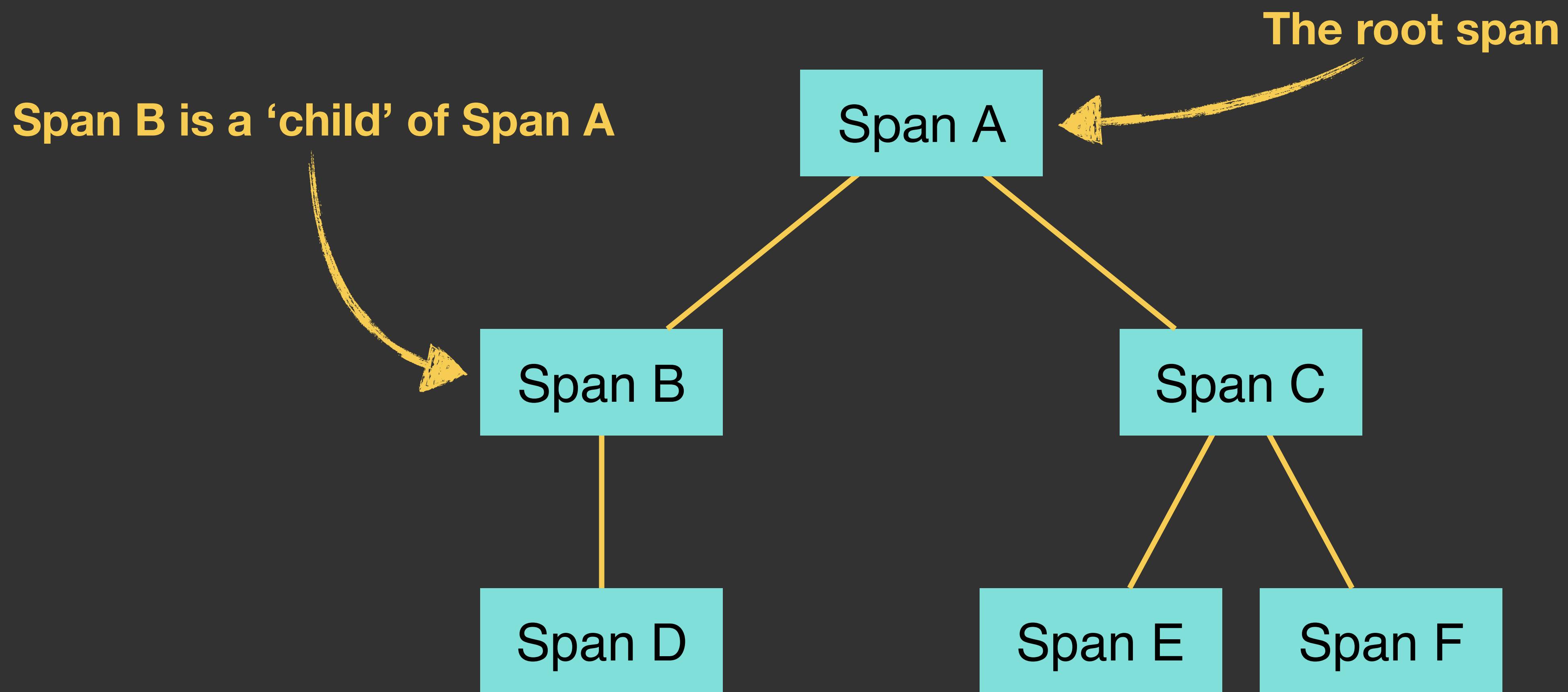


Tracing 101

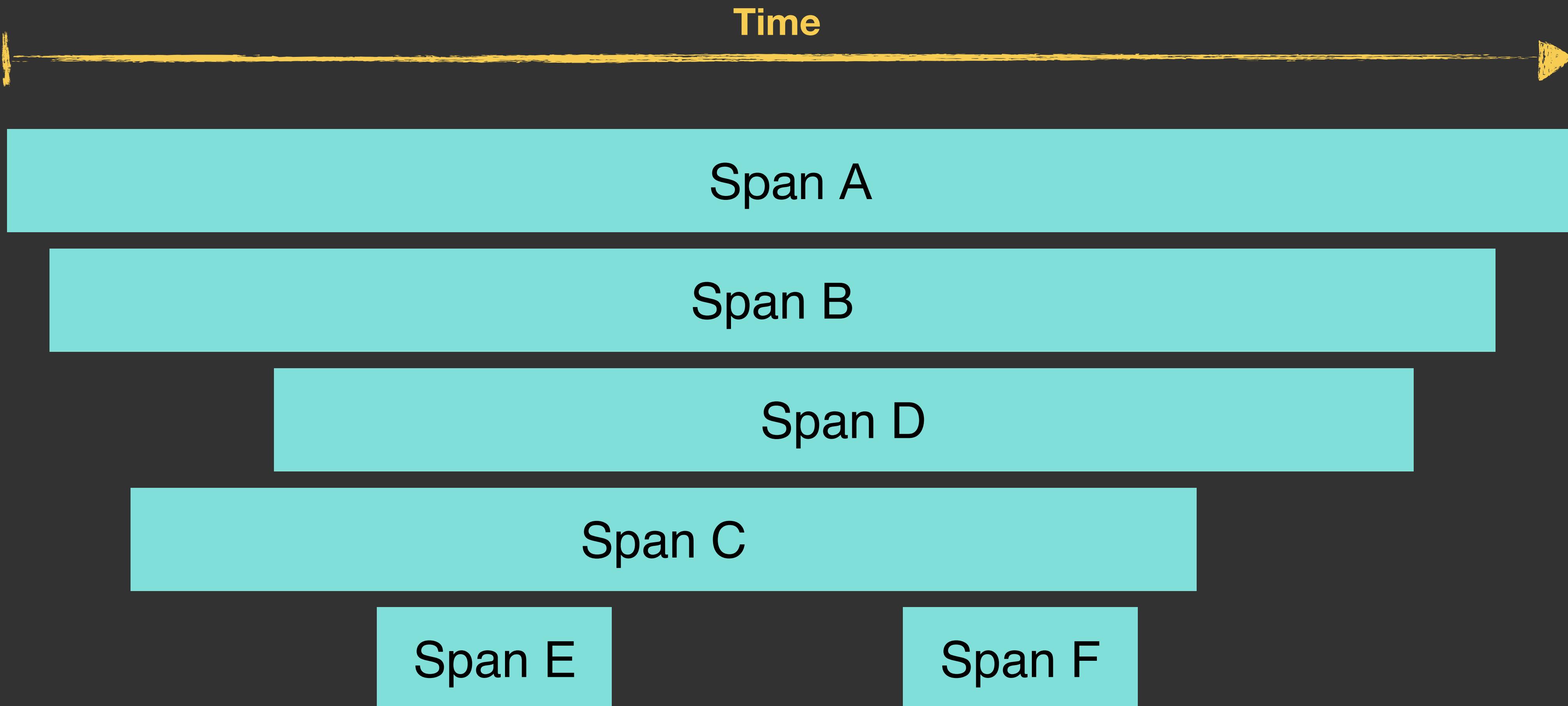


???

Trace Graph



Trace Timeline



```
const span = tracer.startSpan('query', { childOf: otherSpan })  
  
mysql.query(sql, function (err, result) {  
  span.end()  
  // ...  
})
```

```
tracer.startSpan('query')

mysql.query(sql, processResult)

// ...

function processResult (err, result) {
  const span = ???  

  span.end()  

  // ...
}
```



TODO: Get the current span somehow

```
const span = tracer.startSpan('query')

mysql.query(sql, processResult.bind(null, { span }))

// ...

function processResult (ctx, err, result) {
  const span = ctx.span
  span.end()
  // ...
}
```

```
tracer.startSpan('query')

mysql.query(sql, processResult)

// ...

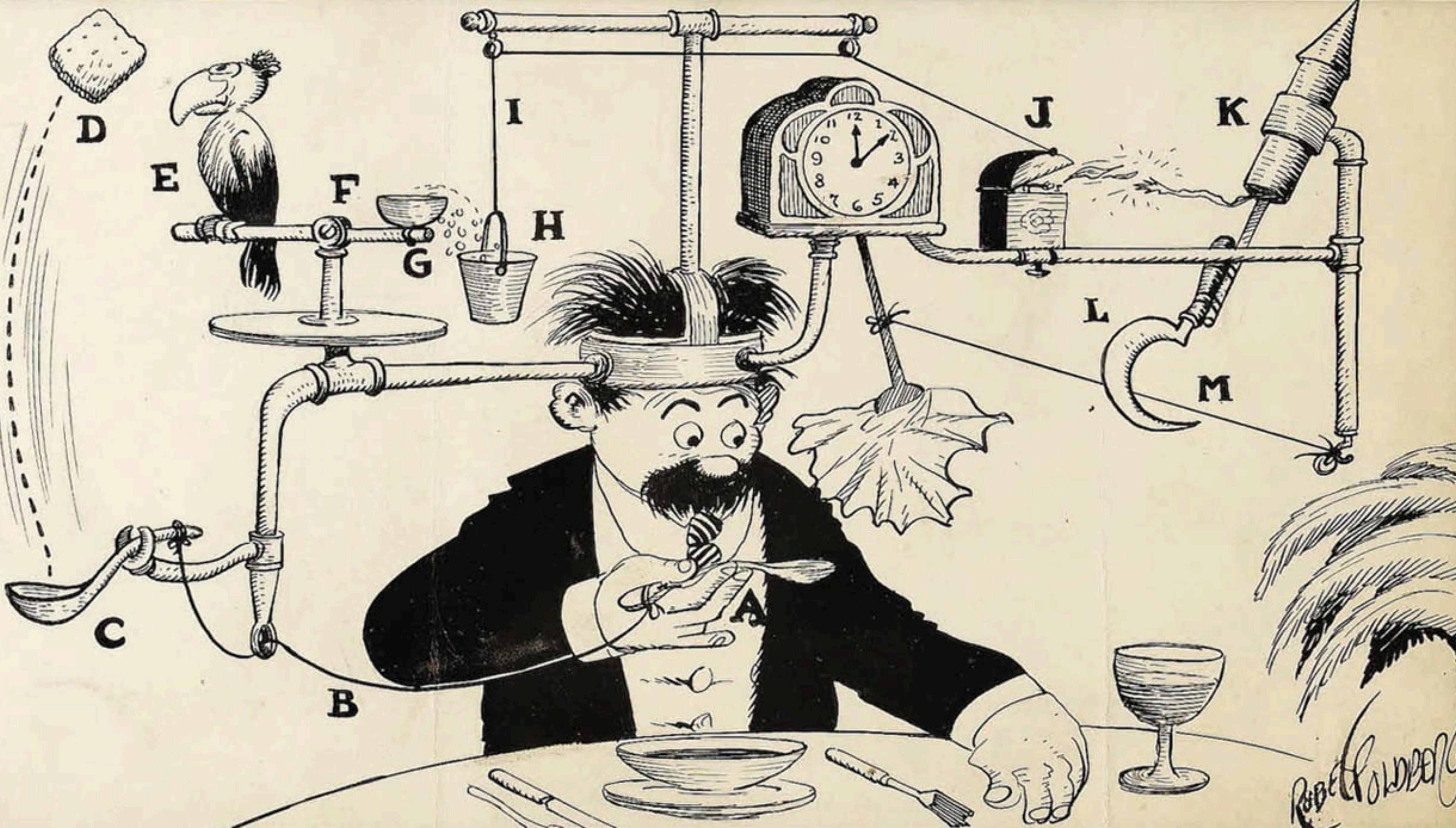
function processResult (err, result) {
  const span = tracer.getCurrentSpan()
  span.end()
  // ...
}
```

```
mysql.query(sql, function (err, result) {  
    // ...  
})
```

```
mysql.qu  
    // ...  
})
```



```
sult) {
```



Dirty Secrets

Patching require
No built-in context propagation
Getting stack traces
Patching every async call -> `async_hooks`

- `async await...` thenables broken
- Promise destroy hook triggered GC
- Uses numeric ids which doesn't play nice with weak maps => hacks in place to clean it up, but those come with risks
- What you want to patch isn't directly exported TAV

Resource management (memory, CPU etc)
Injecting headers into HTTP requests
Babel/TypeScript `defineProperty` hell
Object reuse pool
Metrics: Not a lot of API's to get data from => read things from the filesystem = brittle

How do we know if
mysql is used in the current project?

How do we know when
a mysql query is performed?

`mysql.query()`



1. Start a span when the ‘query’ function is called
2. Associate new span with current trace
3. Make current span child of current “active” span
4. End the span when the query finishes



Patching require()

```
const Module = require('module')
```

```
const Module = require('module')  
  
const module = new Module()
```

```
const Module = require('module')

const module = new Module()

module.require('http')
```

Module.prototype.require

```
const Module = require('module')

const origRequire = Module.prototype.require

Module.prototype.require = function (id) {
  const exports = origRequire.apply(this, arguments)

  if (id === 'http') {
    const request = exports.request

    exports.request = function () {
      const req = request.apply(this, arguments)

      console.log('New outgoing request to', req.getHeader('Host'))

      return req
    }
  }

  return exports
}
```

```
const Module = require('module')

const origRequire = Module.prototype.require

Module.prototype.require = function (id) {
  const exports = origRequire.apply(this, arguments)

  if (id === 'http') {
    const request = exports.request

    exports.request = function () {
      const req = request.apply(this, arguments)

      console.log('New outgoing request to', req.getHeader('Host'))

      return req
    }
  }

  return exports
}
```

```
const Module = require('module')

const origRequire = Module.prototype.require

Module.prototype.require = function (id) {
  const exports = origRequire.apply(this, arguments)

  if (id === 'http') {
    const request = exports.request

    exports.request = function () {
      const req = request.apply(this, arguments)

      console.log('New outgoing request to', req.getHeader('Host'))

      return req
    }
  }

  return exports
}
```

```
const Module = require('module')

const origRequire = Module.prototype.require

Module.prototype.require = function (id) {
  const exports = origRequire.apply(this, arguments)

  if (id === 'http') {
    const request = exports.request

    exports.request = function () {
      const req = request.apply(this, arguments)

      console.log('New outgoing request to', req.getHeader('Host'))

      return req
    }
  }

  return exports
}
```

```
const Module = require('module')

const origRequire = Module.prototype.require

Module.prototype.require = function (id) {
  const exports = origRequire.apply(this, arguments)

  if (id === 'http') {
    const request = exports.request

    exports.request = function () {
      const req = request.apply(this, arguments)

      console.log('New outgoing request to', req.getHeader('Host'))

      return req
    }
  }

  return exports
}
```

```
const Module = require('module')

const origRequire = Module.prototype.require

Module.prototype.require = function (id) {
  const exports = origRequire.apply(this, arguments)

  if (id === 'http') {
    const request = exports.request

    exports.request = function () {
      const req = request.apply(this, arguments)

      console.log('New outgoing request to', req.getHeader('Host'))

      return req
    }
  }

  return exports
}
```

```
const Module = require('module')

const origRequire = Module.prototype.require

Module.prototype.require = function (id) {
  const exports = origRequire.apply(this, arguments)

  if (id === 'http') {
    const request = exports.request

    exports.request = function () {
      const req = request.apply(this, arguments)

      console.log('New outgoing request to', req.getHeader('Host'))

      return req
    }
  }

  return exports
}
```

```
npm install require-in-the-middle
```

```
const path = require('path')
const Hook = require('require-in-the-middle')

// Hook into the express and mongodb module
Hook(['express', 'mongodb'], function (exports, name, basedir) {
  const { version } = require(path.join(basedir, 'package.json'))

  console.log('loading %s@%s', name, version)

  // expose the module version as a property on its exports object
  exports._version = version

  // whatever you return will be returned by `require`
  return exports
})
```

```
const path = require('path')
const Hook = require('require-in-the-middle')

// Hook into the express and mongodb module
Hook(['express', 'mongodb'], function (exports, name, basedir) {
  const { version } = require(path.join(basedir, 'package.json'))

  console.log('loading %s@%s', name, version)

  // expose the module version as a property on its exports object
  exports._version = version

  // whatever you return will be returned by `require`
  return exports
})
```

```
const path = require('path')
const Hook = require('require-in-the-middle')

// Hook into the express and mongodb module
Hook(['express', 'mongodb'], function (exports, name, basedir) {
  const { version } = require(path.join(basedir, 'package.json'))

  console.log('loading %s@%s', name, version)

  // expose the module version as a property on its exports object
  exports._version = version

  // whatever you return will be returned by `require`
  return exports
})
```

```
const path = require('path')
const Hook = require('require-in-the-middle')

// Hook into the express and mongodb module
Hook(['express', 'mongodb'], function (exports, name, basedir) {
  const { version } = require(path.join(basedir, 'package.json'))

  console.log('loading %s@%s', name, version)

  // expose the module version as a property on its exports object
  exports._version = version

  // whatever you return will be returned by `require`
  return exports
})
```

```
const path = require('path')
const Hook = require('require-in-the-middle')

// Hook into the express and mongodb module
Hook(['express', 'mongodb'], function (exports, name, basedir) {
  const { version } = require(path.join(basedir, 'package.json'))

  console.log('loading %s@%s', name, version)

  // expose the module version as a property on its exports object
  exports._version = version

  // whatever you return will be returned by `require`
  return exports
})
```

But wait a second...

ECMAScript Modules 😕

Patching Transpiled Modules or:

How I Learned to Stop Worrying and Love TypeScript & Babel

```
23 * import { parse } from 'graphql';
24 * import { parse } from 'graphql/language';
25 */
26
27 // The GraphQL.js version info.
28 export { version, versionInfo } from './version';
29
30 // The primary entry point into fulfilling a GraphQL request.
31 export { GraphQLArgs, graphql, graphqlSync } from './graphql';
32
33 // Create and operate on GraphQL type definitions and schema.
34 export {
35   // Definitions
36   GraphQLSchema,
37   GraphQLDirective,
38   GraphQLScalarType,
39   GraphQLObjectType,
40   GraphQLInterfaceType,
41   GraphQLUnionType,
42   GraphQLEnumType,
43   GraphQLInputObjectType,
44   GraphQLList,
45   GraphQLNonNull,
46   // Standard GraphQL Scalars
```

```
1 "use strict";
2
3 Object.defineProperty(exports, "__esModule", {
4   value: true
5 });
6 Object.defineProperty(exports, "version", {
7   enumerable: true,
8   get: function get() {
9     return _version.version;
10  }
11 });
12 Object.defineProperty(exports, "versionInfo", {
13   enumerable: true,
14   get: function get() {
15     return _version.versionInfo;
16  }
17 });
18 Object.defineProperty(exports, "graphql", {
19   enumerable: true,
20   get: function get() {
21     return graphql.graphql;
```

```
1 "use strict";
2
3 Object.defineProperty(exports, "__esModule", {
4   value: true
5 });
6 Object.defineProperty(exports, "graphql", {
7   enumerable: true,
8   get: function() {
9     return _v
10    }
11 });
12 Object.defineProperty(exports, "GraphQL", {
13   enumerable: true,
14   get: function() {
15     return _v
16    }
17 });
18 Object.defineProperty(exports, "graphql", {
19   enumerable: true,
20   get: function() {
21     return graphql;
22   }
23 });
24
25 module.exports = exports;
26
27 // This is a test for the issue described in https://github.com/karesu/reflect-metadata/issues/10
28 // It shows that the reflection metadata is not correctly applied to the module's exports object
29 // when it is defined via a function expression.
30 // The problem is that the reflection metadata is only applied to the module's exports object
31 // if it is defined via a function declaration.
32 // This is because the reflection metadata is only applied to the module's exports object
33 // if it is defined via a function declaration.
34 // This is because the reflection metadata is only applied to the module's exports object
35 // if it is defined via a function declaration.
36 // This is because the reflection metadata is only applied to the module's exports object
37 // if it is defined via a function declaration.
38 // This is because the reflection metadata is only applied to the module's exports object
39 // if it is defined via a function declaration.
40 // This is because the reflection metadata is only applied to the module's exports object
41 // if it is defined via a function declaration.
42 // This is because the reflection metadata is only applied to the module's exports object
43 // if it is defined via a function declaration.
44 // This is because the reflection metadata is only applied to the module's exports object
45 // if it is defined via a function declaration.
46 // This is because the reflection metadata is only applied to the module's exports object
47 // if it is defined via a function declaration.
48 // This is because the reflection metadata is only applied to the module's exports object
49 // if it is defined via a function declaration.
50 // This is because the reflection metadata is only applied to the module's exports object
51 // if it is defined via a function declaration.
52 // This is because the reflection metadata is only applied to the module's exports object
53 // if it is defined via a function declaration.
54 // This is because the reflection metadata is only applied to the module's exports object
55 // if it is defined via a function declaration.
56 // This is because the reflection metadata is only applied to the module's exports object
57 // if it is defined via a function declaration.
58 // This is because the reflection metadata is only applied to the module's exports object
59 // if it is defined via a function declaration.
60 // This is because the reflection metadata is only applied to the module's exports object
61 // if it is defined via a function declaration.
62 // This is because the reflection metadata is only applied to the module's exports object
63 // if it is defined via a function declaration.
64 // This is because the reflection metadata is only applied to the module's exports object
65 // if it is defined via a function declaration.
66 // This is because the reflection metadata is only applied to the module's exports object
67 // if it is defined via a function declaration.
68 // This is because the reflection metadata is only applied to the module's exports object
69 // if it is defined via a function declaration.
70 // This is because the reflection metadata is only applied to the module's exports object
71 // if it is defined via a function declaration.
72 // This is because the reflection metadata is only applied to the module's exports object
73 // if it is defined via a function declaration.
74 // This is because the reflection metadata is only applied to the module's exports object
75 // if it is defined via a function declaration.
76 // This is because the reflection metadata is only applied to the module's exports object
77 // if it is defined via a function declaration.
78 // This is because the reflection metadata is only applied to the module's exports object
79 // if it is defined via a function declaration.
80 // This is because the reflection metadata is only applied to the module's exports object
81 // if it is defined via a function declaration.
82 // This is because the reflection metadata is only applied to the module's exports object
83 // if it is defined via a function declaration.
84 // This is because the reflection metadata is only applied to the module's exports object
85 // if it is defined via a function declaration.
86 // This is because the reflection metadata is only applied to the module's exports object
87 // if it is defined via a function declaration.
88 // This is because the reflection metadata is only applied to the module's exports object
89 // if it is defined via a function declaration.
90 // This is because the reflection metadata is only applied to the module's exports object
91 // if it is defined via a function declaration.
92 // This is because the reflection metadata is only applied to the module's exports object
93 // if it is defined via a function declaration.
94 // This is because the reflection metadata is only applied to the module's exports object
95 // if it is defined via a function declaration.
96 // This is because the reflection metadata is only applied to the module's exports object
97 // if it is defined via a function declaration.
98 // This is because the reflection metadata is only applied to the module's exports object
99 // if it is defined via a function declaration.
100 // This is because the reflection metadata is only applied to the module's exports object
101 // if it is defined via a function declaration.
```

```
npm install shallow-clone-shim
```

```
Object.defineProperty(exports, 'foo' {
  enumerable: true,
  get: function get () {
    return 'hello'
  }
})  
  
const clone = require('shallow-clone-shim')  
  
const newExports = clone({}, exports, {
  foo (descriptor) {
    // descriptor == Object.getOwnPropertyDescriptor(exports, 'foo')
    const getter = descriptor.get
    descriptor.get = function get () {
      return getter() + ' world'
    }
    return descriptor
  }
})
```

Patching Async Calls

```
const origSetTimeout = global.setTimeout

global.setTimeout = function (callback, ...args) {
  const origSpan = tracer.currentSpan

  return origSetTimeout(function () {
    const preCallbackSpan = tracer.currentSpan
    tracer.currentSpan = origSpan

    callback.apply(this, arguments)

    tracer.currentSpan = preCallbackSpan
  }, ...args)
}
```

```
const origSetTimeout = global.setTimeout

global.setTimeout = function (callback, ...args) {
  const origSpan = tracer.currentSpan

  return origSetTimeout(function () {
    const preCallbackSpan = tracer.currentSpan
    tracer.currentSpan = origSpan

    callback.apply(this, arguments)

    tracer.currentSpan = preCallbackSpan
  }, ...args)
}
```

```
const origSetTimeout = global.setTimeout

global.setTimeout = function (callback, ...args) {
  const origSpan = tracer.currentSpan

  return origSetTimeout(function () {
    const preCallbackSpan = tracer.currentSpan
    tracer.currentSpan = origSpan

    callback.apply(this, arguments)

    tracer.currentSpan = preCallbackSpan
  }, ...args)
}
```

```
const origSetTimeout = global.setTimeout

global.setTimeout = function (callback, ...args) {
  const origSpan = tracer.currentSpan

  return origSetTimeout(function () {
    const preCallbackSpan = tracer.currentSpan
    tracer.currentSpan = origSpan

    callback.apply(this, arguments)

    tracer.currentSpan = preCallbackSpan
  }, ...args)
}
```

```
const origSetTimeout = global.setTimeout

global.setTimeout = function (callback, ...args) {
  const origSpan = tracer.currentSpan

  return origSetTimeout(function () {
    const preCallbackSpan = tracer.currentSpan
    tracer.currentSpan = origSpan

    callback.apply(this, arguments)

    tracer.currentSpan = preCallbackSpan
  }, ...args)
}
```

```
const origSetTimeout = global.setTimeout

global.setTimeout = function (callback, ...args) {
  const origSpan = tracer.currentSpan

  return origSetTimeout(function () {
    const preCallbackSpan = tracer.currentSpan
    tracer.currentSpan = origSpan

    callback.apply(this, arguments)

    tracer.currentSpan = preCallbackSpan
  }, ...args)
}
```

```
const origSetTimeout = global.setTimeout  
  
global.setTimeout = function (callback, ...args) {  
    const origSpan = tracer.currentSpan  
  
    return origSetTimeout(function () {  
        const preCallbackSpan = tracer.currentSpan  
        tracer.currentSpan = origSpan  
  
        callback.apply(this, arguments)  
  
        tracer.currentSpan = preCallbackSpan  
    }, ...args)  
}
```

All async API's

- net
- http
- child_process
- timers
- dns
- fs
- zlib
- crypto
- global.Promise

```
require('async_hooks')
```

But wait a second...

Thenables 😢

github.com / nodejs / node / issues / 22360

The screenshot shows a GitHub issue page for the Node.js repository. The title of the issue is "Async Hooks do not recognize execution contexts created when processing thenables". The issue was opened by medikoo on August 16, 2018, with 54 comments. The issue has 883 issues, 360 pull requests, 3 actions, 3 projects, and 15.5k forks.

Issue Details:

- Title:** Async Hooks do not recognize execution contexts created when processing thenables #22360
- Status:** Open
- Comments:** 54
- Watchers:** 2.9k
- Unstar:** 65.9k
- Forks:** 15.5k

Comments:

medikoo commented on 16 Aug 2018 • edited

Promise.all to observe input promises, invokes promise.then in next event loop, at least it's what stack traces show in both Node.js and Google Chrome.

Still Async Hooks see it as same execution context, it can be seen by running following code:

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

async_hooks.createHook({ init() {} }).enable();

Promise.all([
  {
    then() {
      console.log(async_hooks.executionAsyncId(), async_hooks.triggerAsyncId(), 'thenable');
      console.trace('thenable.then invoked by Promise.all');
    },
  },
]);
```

Assignees: No one—assign yourself

Labels: **async_hooks**, feature request

Projects: None yet

Milestone: No milestone

Resource Usage

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

const currentSpans = new Map()

asyncHooks.createHook({
  init (asyncId) {
    const span = tracer.currentSpan

    if (span) {
      currentSpans.set(asyncId, span)
    }
  },
  destroy (asyncId) {
    currentSpans.delete(asyncId)
  }
})
```

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

const currentSpans = new Map()

asyncHooks.createHook({
  init (asyncId) {
    const span = tracer.currentSpan

    if (span) {
      currentSpans.set(asyncId, span)
    }
  },
  destroy (asyncId) {
    currentSpans.delete(asyncId)
  }
})
```

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

const currentSpans = new Map()

asyncHooks.createHook({
  init (asyncId) {
    const span = tracer.currentSpan

    if (span) {
      currentSpans.set(asyncId, span)
    }
  },
  destroy (asyncId) {
    currentSpans.delete(asyncId)
  }
})
```

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

const currentSpans = new Map()

asyncHooks.createHook({
  init (asyncId) {
    const span = tracer.currentSpan

    if (span) {
      currentSpans.set(asyncId, span)
    }
  },
  destroy (asyncId) {
    currentSpans.delete(asyncId)
  }
})
```

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

const currentSpans = new Map()

asyncHooks.createHook({
  init (asyncId) {
    const span = tracer.currentSpan

    if (span) {
      currentSpans.set(asyncId, span)
    }
  },
  destroy (asyncId) {
    currentSpans.delete(asyncId)
  }
})
```

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

const currentSpans = new Map()

asyncHooks.createHook({
  init (asyncId) {
    const span = tracer.currentSpan

    if (span) {
      currentSpans.set(asyncId, span)
    }
  },
  destroy (asyncId) {
    currentSpans.delete(asyncId)
  }
})
```

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

```
const currentSpans = new Map()
```

```
asyncHooks.createHook({  
  init (asyncId) {  
    const span = tracer.currentSpan
```

Can we trust (when)
destroy is called?

```
    if (span) {  
      currentSpans.set(asyncId, span)  
    }  
  },  
  
  destroy (asyncId) {  
    currentSpans.delete(asyncId)  
  }  
})
```

Not a WeakMap 😢

Userland Callback Queues

```
class Client {  
  constructor () {  
    this.pool = new Pool({ size: 5 })  
    this.queue = []  
  }  
  
  request (query, callback) {  
    const socket = this.pool.getAvailableSocket()  
  
    if (socket) {  
      socket.send(query, (err, res) => {  
        callback(err, res)  
  
        if (this.queue.length > 0) {  
          const [query, callback] = this.queue.shift()  
          this.request(query, callback)  
        }  
      })  
    } else {  
      this.queue.push([query, callback])  
    }  
  }  
}
```

```
class Client {  
  constructor () {  
    this.pool = new Pool({ size: 5 })  
    this.queue = []  
  }  
  
  request (query, callback) {  
    const socket = this.pool.getAvailableSocket()  
  
    if (socket) {  
      socket.send(query, (err, res) => {  
        callback(err, res)  
  
        if (this.queue.length > 0) {  
          const [query, callback] = this.queue.shift()  
          this.request(query, callback)  
        }  
      })  
    } else {  
      this.queue.push([query, callback])  
    }  
  }  
}
```

```
class Client {  
  constructor () {  
    this.pool = new Pool({ size: 5 })  
    this.queue = []  
  }  
  
  request (query, callback) {  
    const socket = this.pool.getAvailableSocket()  
  
    if (socket) {  
      socket.send(query, (err, res) => {  
        callback(err, res)  
  
        if (this.queue.length > 0) {  
          const [query, callback] = this.queue.shift()  
          this.request(query, callback)  
        }  
      })  
    } else {  
      this.queue.push([query, callback])  
    }  
  }  
}
```

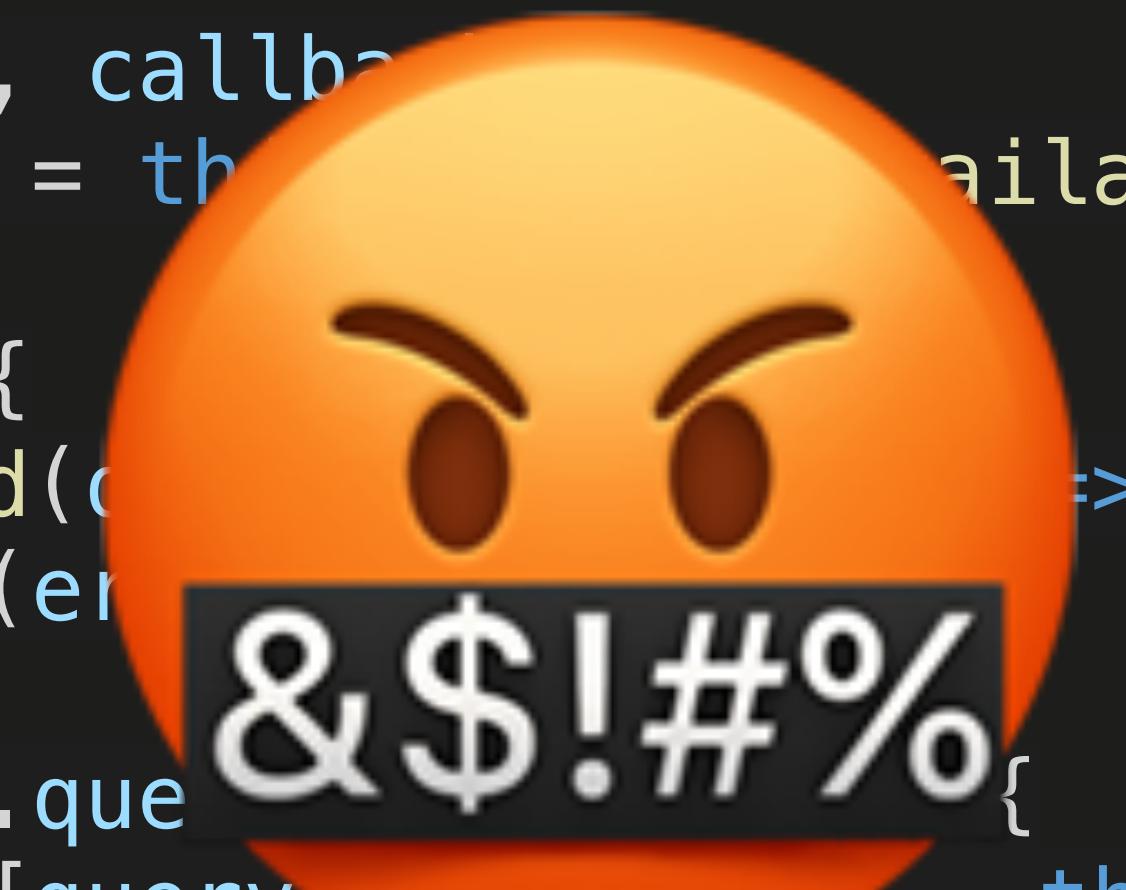
```
class Client {  
  constructor () {  
    this.pool = new Pool({ size: 5 })  
    this.queue = []  
  }  
  
  request (query, callback) {  
    const socket = this.pool.getAvailableSocket()  
  
    if (socket) {  
      socket.send(query, (err, res) => {  
        callback(err, res)  
  
        if (this.queue.length > 0) {  
          const [query, callback] = this.queue.shift()  
          this.request(query, callback)  
        }  
      })  
    } else {  
      this.queue.push([query, callback])  
    }  
  }  
}
```

```
class Client {  
  constructor () {  
    this.pool = new Pool({ size: 5 })  
    this.queue = []  
  }  
  
  request (query, callback) {  
    const socket = this.pool.getAvailableSocket()  
  
    if (socket) {  
      socket.send(query, (err, res) => {  
        callback(err, res)  
  
        if (this.queue.length > 0) {  
          const [query, callback] = this.queue.shift()  
          this.request(query, callback)  
        }  
      })  
    } else {  
      this.queue.push([query, callback])  
    }  
  }  
}
```

```
class Client {  
  constructor () {  
    this.pool = new Pool({ size: 5 })  
    this.queue = []  
  }  
  
  request (query, callback) {  
    const socket = this.pool.getAvailableSocket()  
  
    if (socket) {  
      socket.send(query, (err, res) => {  
        callback(err, res)  
  
        if (this.queue.length > 0) {  
          const [query, callback] = this.queue.shift()  
          this.request(query, callback)  
        }  
      })  
    } else {  
      this.queue.push([query, callback])  
    }  
  }  
}
```

Called in the context
of the previous
'socket.send()' call

```
class Client {  
  constructor () {  
    this.pool = new Pool({ size: 5 })  
    this.queue = []  
  }  
  
  request (query, callback) {  
    const socket = this.pool.getAvailableSocket()  
  
    if (socket) {  
      socket.send(query)  
      return callback(null)  
    } else {  
      if (this.queue.length) {  
        const [query, callback] = this.queue.shift()  
        this.request(query, callback)  
      }  
    }  
  }  
}
```



```
class Client {  
  constructor () {  
    this.pool = new Pool({ size: 5 })  
  }
```

Which is called in
the context of the
'socket.send()' call

This.currentSpan turns to null after a POST request #515



rannygmx opened this issue on 15 Aug 2018 · 20 comments

Called in the context
of the previous
'socket.send()' call

```
const socket = this.pool.getAvailableSocket()  
  
if (socket) {  
  socket.send(query, (err, res) => {  
    callback(err, res)  
  
    if (this.queue.length > 0) {  
      const [query, callback] = this.queue.shift()  
      this.request(query, callback)  
    }  
  })  
} else {  
  this.queue.push([query, callback])  
}  
}
```



watson commented on 15 Aug 2018

Member

+ 😊 ...

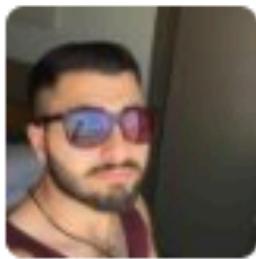
Hi @rannygmx

Thanks for letting us know about this issue. The problem you're experiencing is most likely related to a context propagation issue. This happens when your application performs an async operation that we didn't anticipate. This is when a callback is scheduled to be called after some async operation finishes, but we fail to register it. If so, we don't know which transaction was the active transaction when the callback is finally called, and so `currentTransaction` is suddenly `null`.

90% of the time this is because of what's called a "user-land callback queue", where for instance a database-driver maintains an internal pool of connections to the databases. When you then make a database query, but all connections in the pool are in use, your query is put into a queue that waits until one of the connections in the pool becomes available. If the Node.js agent doesn't know about this pool, we'll also not know which transaction should be activated once the result of the query is returned.

Step one in figuring out what's going on in your particular case is to take a look at your dependencies. If you don't mind, could you please share with us the list of dependencies in your `package.json` file?

```
class Client {
```



rannygmx commented on 15 Aug 2018

Author + 😊 ...

Thanks for the explanation,
This is my dependencies list:

```
"dependencies": {  
  "async": "^2.4.1",  
  "body-parser": "^1.12.3",  
  "compression": "^1.4.3",  
  "cors": "^2.6.0",  
  "cron": "^1.2.1",  
  "dom-js": "^0.0.9",  
  "elastic-apm-node": "^1.7.0",  
  "elasticsearch": "^13.1.1",  
  "errorhandler": "^1.3.5",  
  "express": "^4.12.3",  
  "kafka-node": "^0.3.2",  
  "lodash": "^4.13.1",  
  "method-override": "^2.3.2",
```



Test Dependency Hell

```
npm install test-all-versions
```

```
py:
  versions: '>=4 <8'
  peerDependencies:
    - bluebird@^3.0.0
    - knex@^0.17.3
  commands:
    - node test/instrumentation/modules/pg/pg.js
    - node test/instrumentation/modules/pg/knex.js
mongodb-core:
  versions: '>=1.2.19 <4'
  commands: node test/instrumentation/modules/mongodb-core.js
mongodb:
  versions: '>=3.3'
  commands: node test/instrumentation/modules/mongodb.js
bluebird:
  versions: '>=2 <4'
  commands:
    - node test/instrumentation/modules/bluebird/bluebird.js
    - node test/instrumentation/modules/bluebird/control.js
```


Interservice Communication

GET /products HTTP/1.1
Host: www.example.com 
Date: Mon, 29 Oct 2018 16:11:05 GMT
Connection: keep-alive
Content-Length: 0

<magic header>

```
GET /products HTTP/1.1
Host: www.example.com
<magic header>
Date: Mon, 29 Oct 2018 16:11:05 GMT
Connection: keep-alive
Content-Length: 0
```

GET /products HTTP/1.1

Host: www.example.com

traceparent: 00-82c5500f40667e5500e9ae8e9711553c-992631f881f78c3b-01

Date: Mon, 29 Oct 2018 16:11:05 GMT

Connection: keep-alive

Content-Length: 0

GET /products HTTP/1.1
Host: www.example.com
traceparent: 00-82c550e0-992631f881f78c3b-01
Date: Mon, 29 Oct 2018
Connection: keep-alive
Content-Length: 0

Candidate
Recommendation



Trace Context Working Group

GET /products HTTP/1.1

Host: www.example.com

traceparent: 00-82c5500f40667e5500e9ae8e9711553c-992631f881f78c3b-01

Date: Mon, 29 Oct 2018 16:11:05 GMT

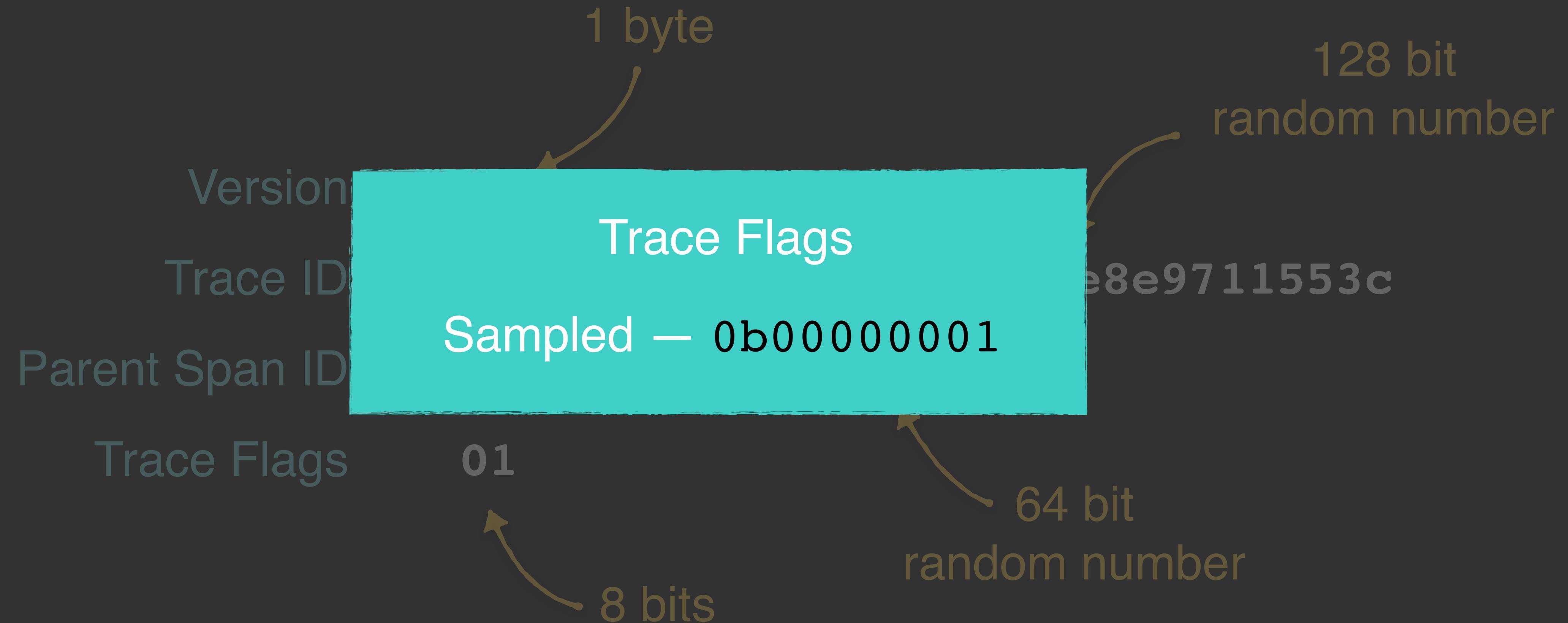
Connection: keep-alive

Content-Length: 0

traceparent



traceparent



github.com / w3c / trace-context-binary

The screenshot shows a GitHub repository page for the 'w3c/trace-context-binary' project. The repository has 9 commits, 2 branches, 0 releases, and 2 contributors. The latest commit was made 22 days ago. The repository URL is <https://w3c.github.io/trace-context-b...>.

Code | Issues 4 | Pull requests 0 | Insights

9 commits | 2 branches | 0 releases | 2 contributors | View license

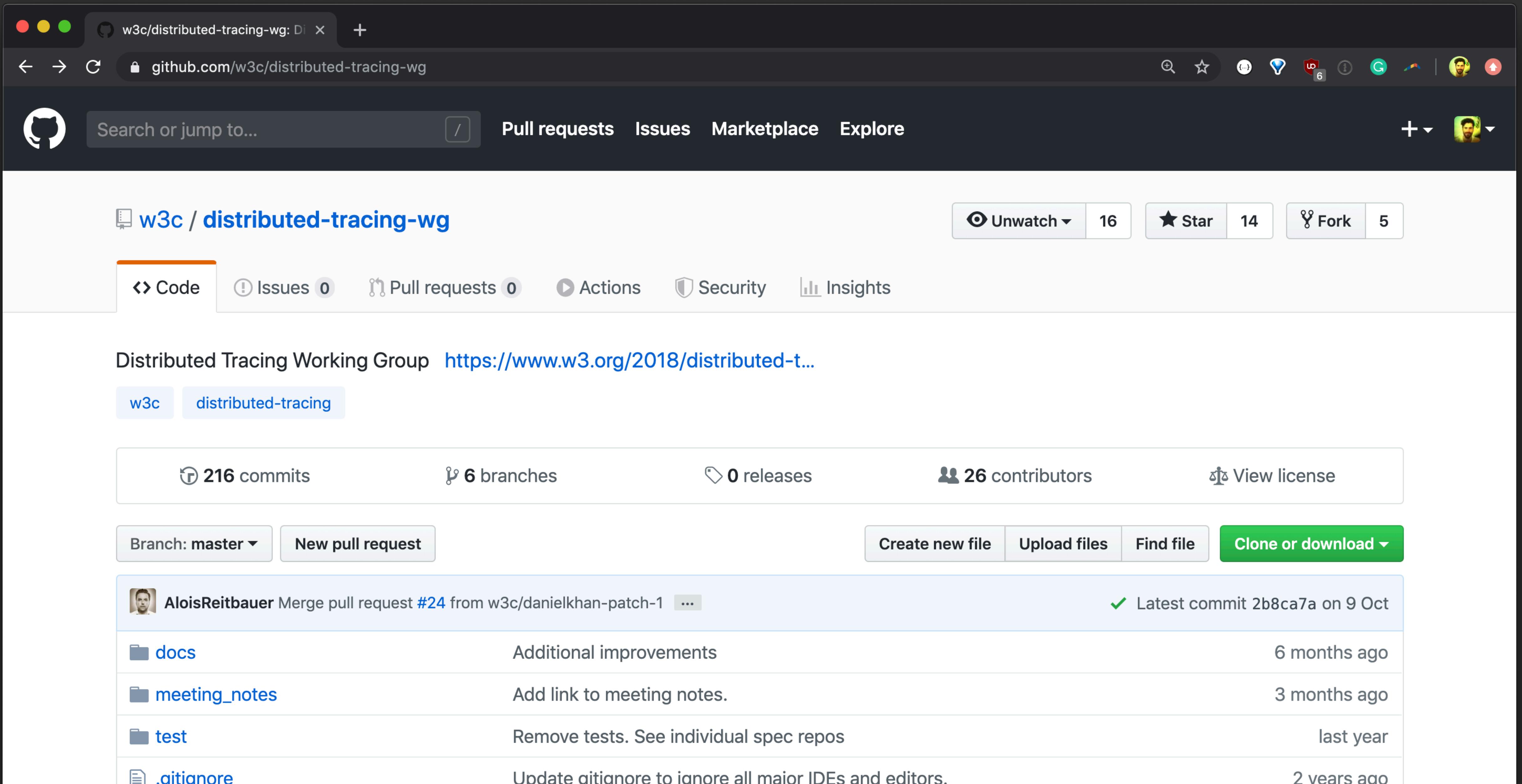
Branch: master | New pull request | Create new file | Upload files | Find File | Clone or download

SergeyKanzhelev Initial proposal for binary protocol (#2) ... | Latest commit 571cafa 22 days ago

File	Description	Time
spec	Initial proposal for binary protocol (#2)	22 days ago
CODE_OF_CONDUCT.md	Adding baseline CODE_OF_CONDUCT.md	3 months ago
CONTRIBUTING.md	Moved from w3c/trace-context	3 months ago
LICENSE.md	Moved from w3c/trace-context	3 months ago
README.md	Moved from w3c/trace-context	3 months ago
index.html	Initial proposal for binary protocol (#2)	22 days ago

Standards

github.com / w3c / distributed-tracing-wg



The screenshot shows the GitHub repository page for `w3c/distributed-tracing-wg`. The repository name is displayed at the top left. The header includes a search bar, navigation links for Pull requests, Issues, Marketplace, and Explore, and user profile icons. Below the header, the repository name is shown again with a book icon. To the right are buttons for Unwatch (16), Star (14), and Fork (5). A navigation bar below the header includes tabs for Code (selected), Issues (0), Pull requests (0), Actions, Security, and Insights.

Distributed Tracing Working Group <https://www.w3.org/2018/distributed-t...>

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

Icon	Commit Details	File Changes	Last Commit
	AloisReitbauer Merge pull request #24 from w3c/danielkhan-patch-1	...	✓ Latest commit 2b8ca7a on 9 Oct
	docs	Additional improvements	6 months ago
	meeting_notes	Add link to meeting notes.	3 months ago
	test	Remove tests. See individual spec repos	last year
	.gitignore	Update gitignore to ignore all major IDEs and editors.	2 years ago

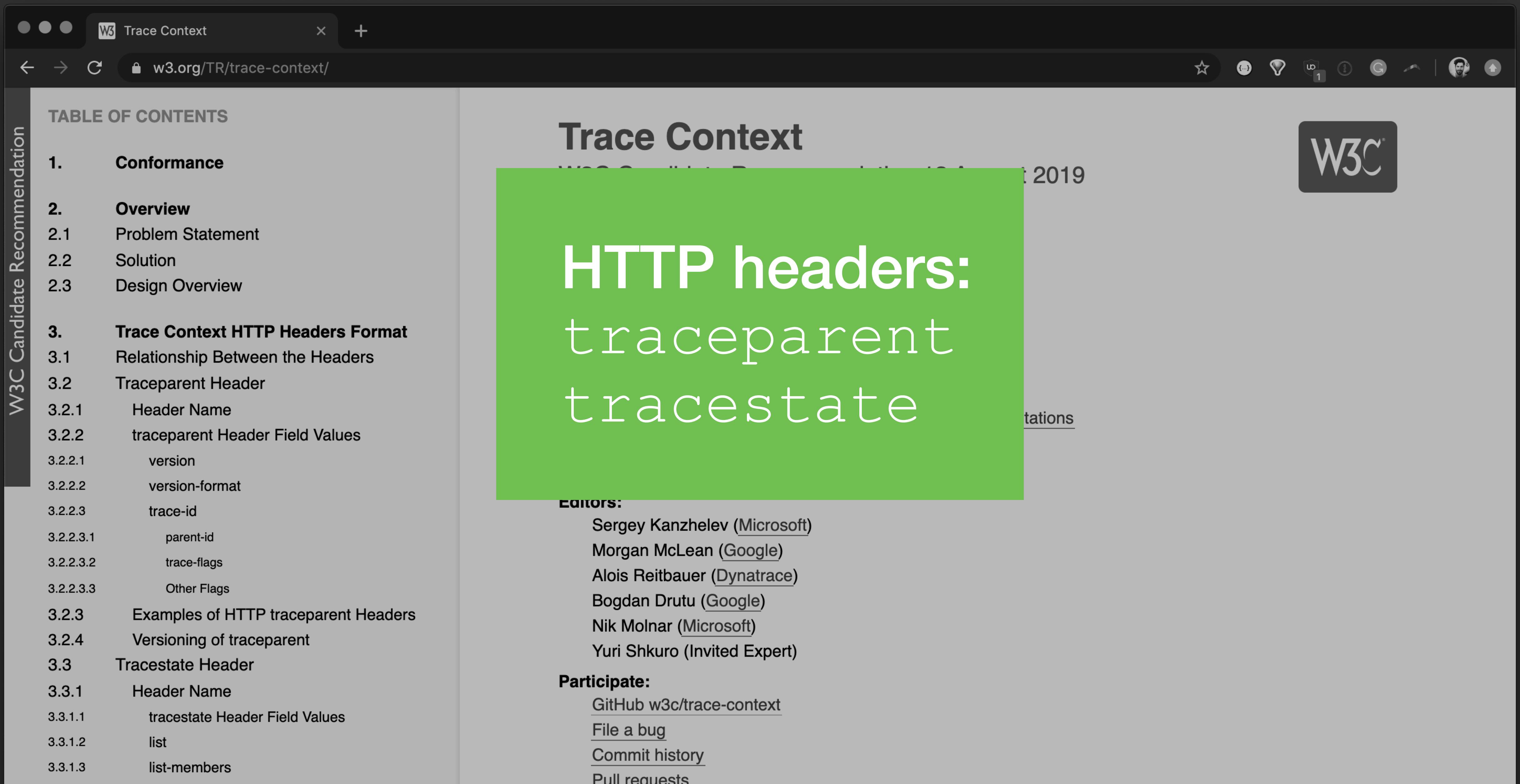
The screenshot shows a web browser window with the following details:

- Title Bar:** W3 Trace Context
- Address Bar:** w3.org/TR/trace-context/
- Toolbar:** Standard browser icons for back, forward, search, and refresh.
- W3C Candidate Recommendation Sidebar:** A vertical sidebar on the left with the title "W3C Candidate Recommendation".
- Table of Contents:** A list of sections and sub-sections for the Trace Context specification.
- Main Content Area:** The title "Trace Context" in large blue font, followed by "W3C Candidate Recommendation 13 August 2019".
- W3C Logo:** A blue square logo with the white text "W3C".
- Information Links:** Various links for this version, latest published version, latest editor's draft, implementation report, previous version, editors, and participate options.

Table of Contents (Left Sidebar):

- 1. Conformance
- 2. Overview
 - 2.1 Problem Statement
 - 2.2 Solution
 - 2.3 Design Overview
- 3. Trace Context HTTP Headers Format
 - 3.1 Relationship Between the Headers
 - 3.2 Traceparent Header
 - 3.2.1 Header Name
 - 3.2.2 traceparent Header Field Values
 - 3.2.2.1 version
 - 3.2.2.2 version-format
 - 3.2.2.3 trace-id
 - 3.2.2.3.1 parent-id
 - 3.2.2.3.2 trace-flags
 - 3.2.2.3.3 Other Flags
 - 3.2.3 Examples of HTTP traceparent Headers
 - 3.2.4 Versioning of traceparent
 - 3.3 Tracestate Header
 - 3.3.1 Header Name
 - 3.3.2 tracestate Header Field Values
 - 3.3.2.1 list
 - 3.3.2.2 list-members

w3.org / TR / trace-context



The screenshot shows a web browser displaying the W3C Trace Context specification page. The title bar reads "W3 Trace Context". The URL in the address bar is "w3.org/TR/trace-context/". The page content includes a "TABLE OF CONTENTS" on the left and a main "Trace Context" section on the right. The main section features a large green banner with the text "HTTP headers: traceparent tracestate". It also lists "Editors" and "Participate" links.

W3C Candidate Recommendation

TABLE OF CONTENTS

- 1. Conformance
- 2. Overview
 - 2.1 Problem Statement
 - 2.2 Solution
 - 2.3 Design Overview
- 3. Trace Context HTTP Headers Format
 - 3.1 Relationship Between the Headers
 - 3.2 Traceparent Header
 - 3.2.1 Header Name
 - 3.2.2 traceparent Header Field Values
 - 3.2.2.1 version
 - 3.2.2.2 version-format
 - 3.2.2.3 trace-id
 - 3.2.2.3.1 parent-id
 - 3.2.2.3.2 trace-flags
 - 3.2.2.3.3 Other Flags
 - 3.2.3 Examples of HTTP traceparent Headers
 - 3.2.4 Versioning of traceparent
 - 3.3 Tracestate Header
 - 3.3.1 Header Name
 - 3.3.1.1 tracestate Header Field Values
 - 3.3.1.2 list
 - 3.3.1.3 list-members

W3C Editor's Draft

Propagation format for distributed trace context: Trace Context headers

W3C Editor's Draft 16 May 2019

This version: <https://w3c.github.io/correlation-context/>

Latest published version: <https://www.w3.org/TR/correlation-context/>

Latest editor's draft: <https://w3c.github.io/correlation-context/>

Editors:

Sergey Kanzhelev ([Microsoft](#))
Morgan McLean ([Google](#))
Alois Reitbauer ([Dynatrace](#))

Participate:

[GitHub w3c/correlation-context](#)
[File a bug](#)
[Commit history](#)
[Pull requests](#)

Discussions:

[We are on Gitter.](#)

Copyright © 2019 W3C® ([MIT](#), [ERCIM](#), [Keio](#), [Beihang](#)). W3C [liability](#), [trademark](#) and [permissive document license](#) rules apply.

w3c.github.io / correlation-context

W3C Editors Draft

W3 Propagation format for distribu X +
w3c.github.io/correlation-context/ 

TABLE OF CONTENTS

1. Overview
2. Correlation Context HTTP Header Format
 - 2.1 Format
 - 2.1.1 Header name
 - 2.1.2 Header value
 - 2.1.3 Name format
 - 2.1.4 Value format
 - 2.1.5 Properties
 - 2.2 Examples of HTTP headers
 - 2.2.1 Example use case

Propagation format for distributed trace context: Trace Context headers  W3C Editor's Draft 6 MAY 2019 

HTTP headers: correlation-context

Sergey Kanzhelev ([Microsoft](#))

Morgan McLean ([Google](#))

Alois Reitbauer ([Dynatrace](#))

Participate:

[GitHub w3c/correlation-context](#)

[File a bug](#)

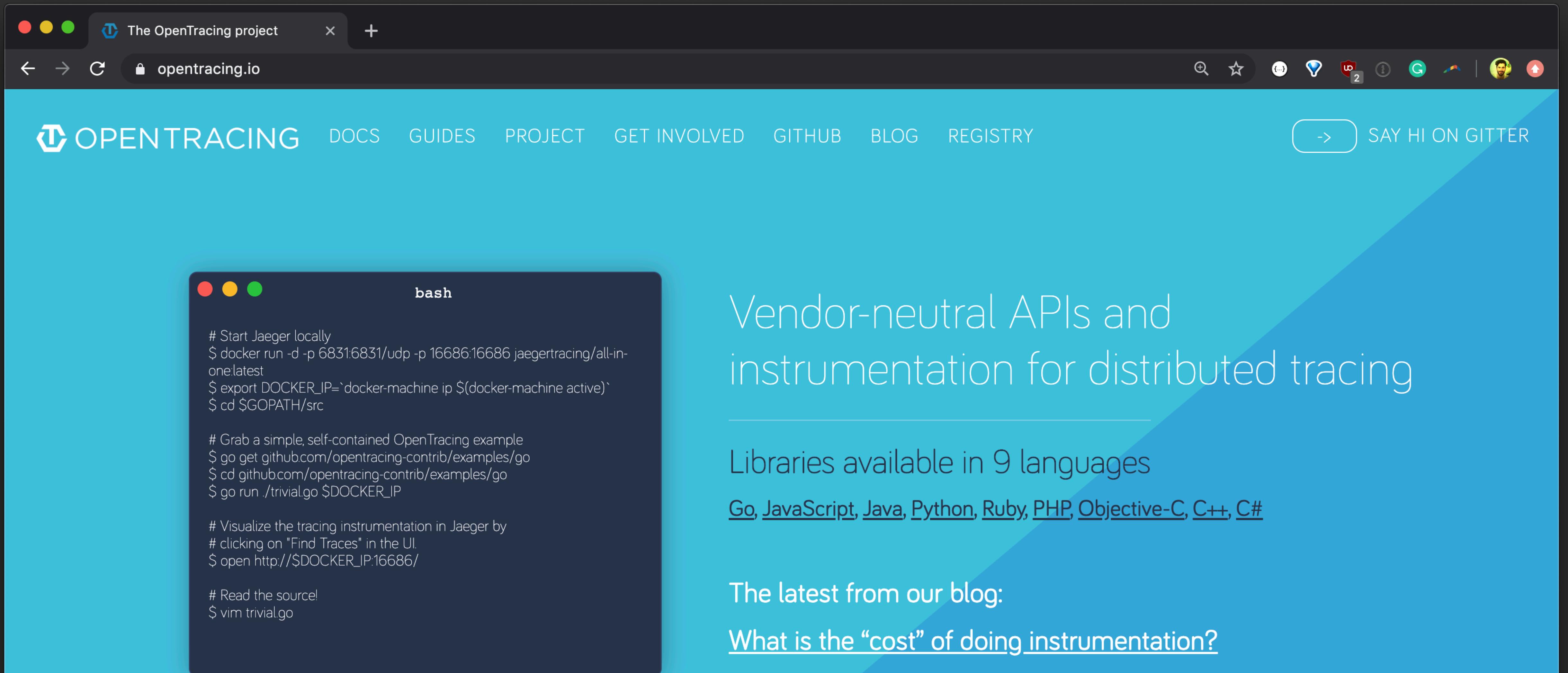
[Commit history](#)

[Pull requests](#)

Discussions:

[We are on Gitter.](#)

opentracing.io



The screenshot shows a web browser window with the title bar "The OpenTracing project" and the URL "opentracing.io". The page itself has a teal gradient background. At the top, there's a navigation bar with links for "OPENTRACING", "DOCS", "GUIDES", "PROJECT", "GET INVOLVED", "GITHUB", "BLOG", and "REGISTRY". On the right, there's a button "SAY HI ON GITTER" with a "->" icon. Below the navigation, there's a large text area with a dark background containing a terminal session titled "bash". The terminal session shows commands to start Jaeger locally, clone an OpenTracing example from GitHub, run it, and open a browser to visualize the tracing instrumentation. To the right of this terminal box, the main content area features the text "Vendor-neutral APIs and instrumentation for distributed tracing" and "Libraries available in 9 languages" followed by a list of supported languages: Go, JavaScript, Java, Python, Ruby, PHP, Objective-C, C++, and C#. Further down, it says "The latest from our blog:" and lists an article: "What is the “cost” of doing instrumentation?".

```
# Start Jaeger locally
$ docker run -d -p 6831:6831/udp -p 16686:16686 jaegertracing/all-in-one:latest
$ export DOCKER_IP=`docker-machine ip $(docker-machine active)`
$ cd $GOPATH/src

# Grab a simple, self-contained OpenTracing example
$ go get github.com/opentracing-contrib/examples/go
$ cd github.com/opentracing-contrib/examples/go
$ go run ./trivial.go $DOCKER_IP

# Visualize the tracing instrumentation in Jaeger by
# clicking on "Find Traces" in the UI.
$ open http://$DOCKER_IP:16686/

# Read the source!
$ vim trivial.go
```

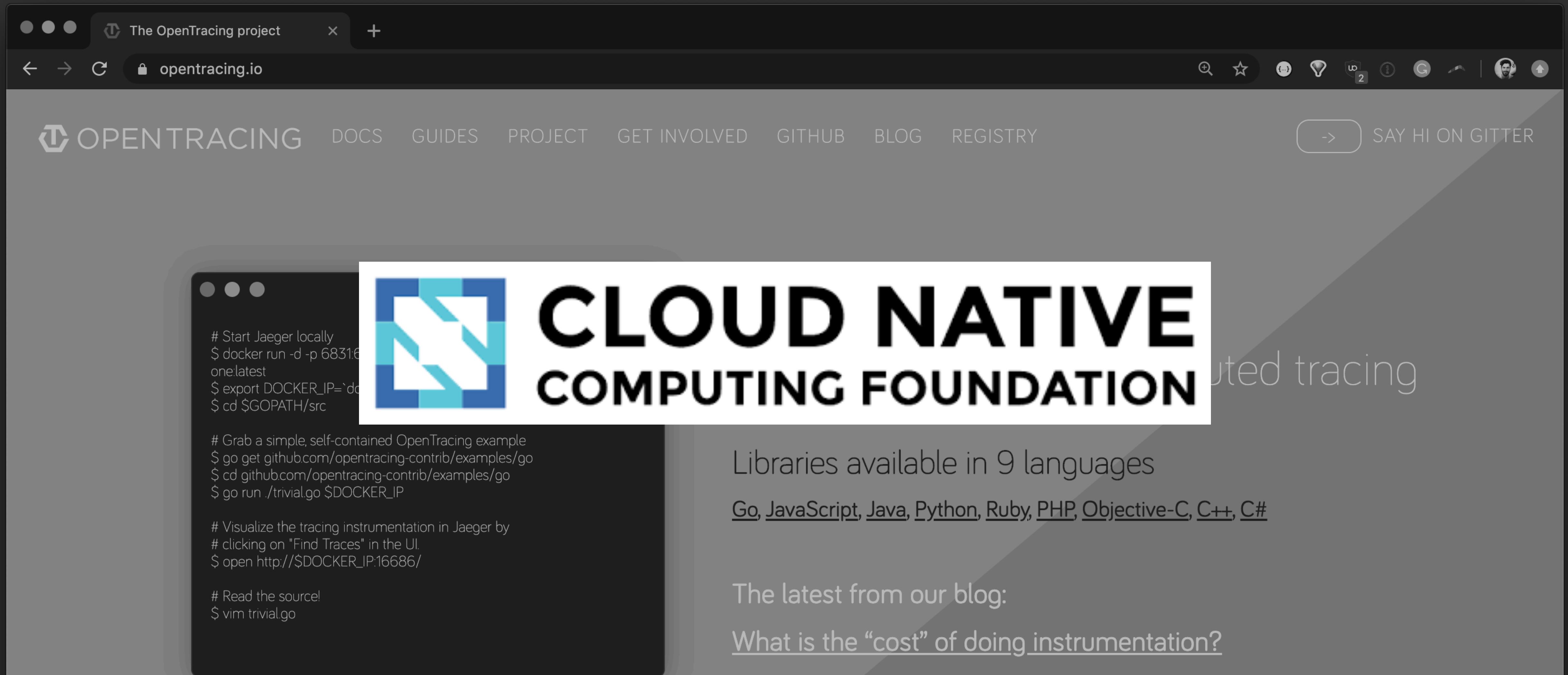
Vendor-neutral APIs and
instrumentation for distributed tracing

Libraries available in 9 languages

[Go](#), [JavaScript](#), [Java](#), [Python](#), [Ruby](#), [PHP](#), [Objective-C](#), [C++](#), [C#](#)

The latest from our blog:
[What is the “cost” of doing instrumentation?](#)

opentracing.io



The OpenTracing project

opentracing.io

OPEN TRACING DOCS GUIDES PROJECT GET INVOLVED GITHUB BLOG REGISTRY -> SAY HI ON GITTER

Start Jaeger locally
\$ docker run -d -p 6831:6831 jaeger/one:latest
\$ export DOCKER_IP=`docker inspect --format='{{.NetworkSettings.IPAddress}}' \$(docker ps -q)`
\$ cd \$GOPATH/src

Grab a simple, self-contained OpenTracing example
\$ go get github.com/opentracing-contrib/examples/go
\$ cd github.com/opentracing-contrib/examples/go
\$ go run ./trivial.go \$DOCKER_IP

Visualize the tracing instrumentation in Jaeger by
clicking on "Find Traces" in the UI.
\$ open http://\$DOCKER_IP:16686/

Read the source!
\$ vim trivial.go

CLOUD NATIVE COMPUTING FOUNDATION

Cloud Native Computing Foundation

Libraries available in 9 languages

Go, [JavaScript](#), [Java](#), [Python](#), [Ruby](#), [PHP](#), [Objective-C](#), [C++](#), [C#](#)

The latest from our blog:

[What is the “cost” of doing instrumentation?](#)

opentracing.io

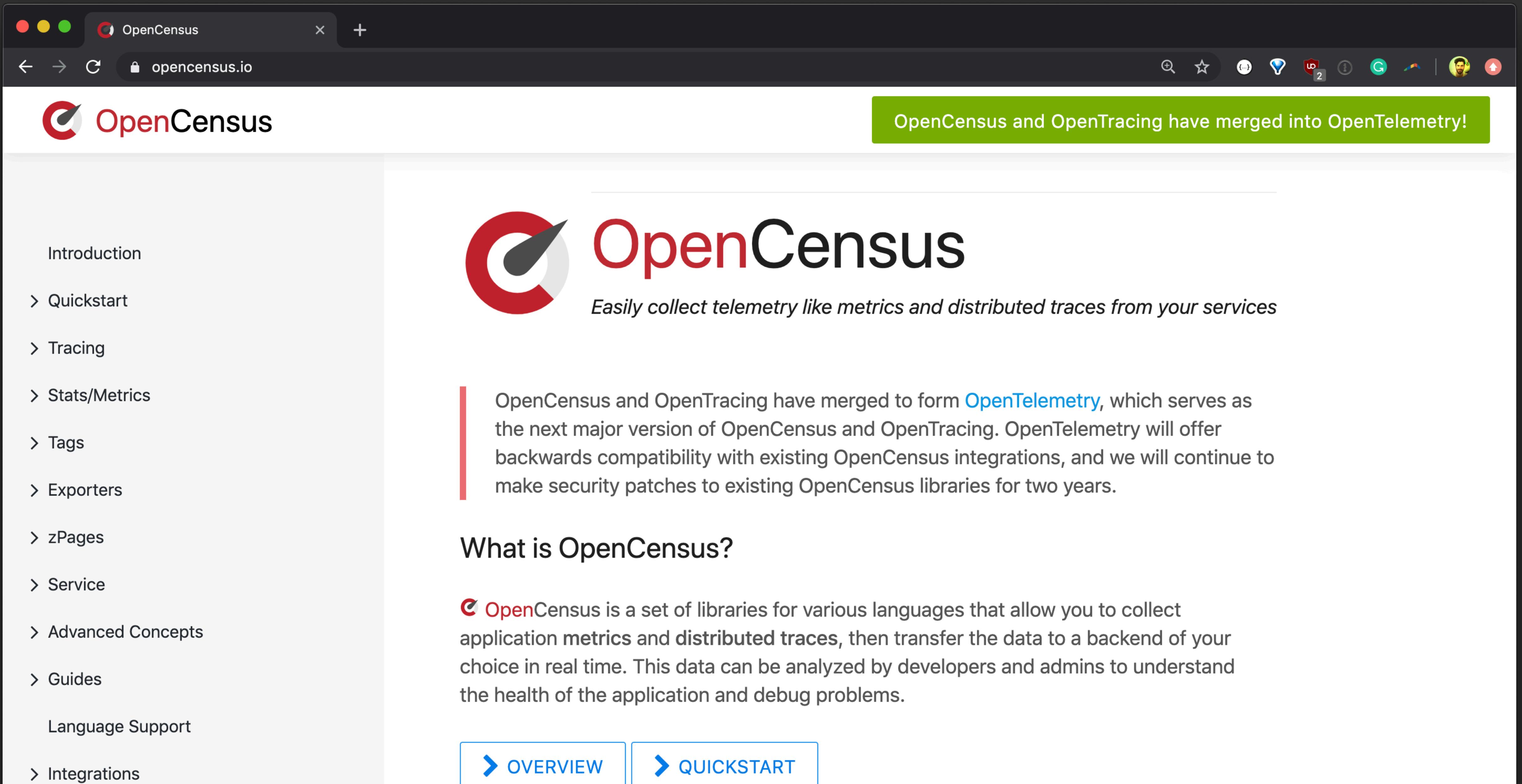
The screenshot shows a browser window with the URL opentracing.io in the address bar. The page content is a code snippet demonstrating how to use the OpenTracing library to log an error event. The code uses the tracer's startSpan method to begin a span, sets an error tag on it, logs an error event with the error object, message, and stack trace, and finally finishes the span.

```
const span = tracer.startSpan('http_request')

span.setTag(opentracing.Tags.ERROR, true)

span.log({
  event: 'error',
  'error.object': err,
  message: err.message,
  stack: err.stack
})

span.finish()
```



The screenshot shows a web browser window with the URL opencensus.io in the address bar. The page content is as follows:

OpenCensus

OpenCensus and OpenTracing have merged into OpenTelemetry!

OpenCensus
Easily collect telemetry like metrics and distributed traces from your services

OpenCensus and OpenTracing have merged to form [OpenTelemetry](#), which serves as the next major version of OpenCensus and OpenTracing. OpenTelemetry will offer backwards compatibility with existing OpenCensus integrations, and we will continue to make security patches to existing OpenCensus libraries for two years.

What is OpenCensus?

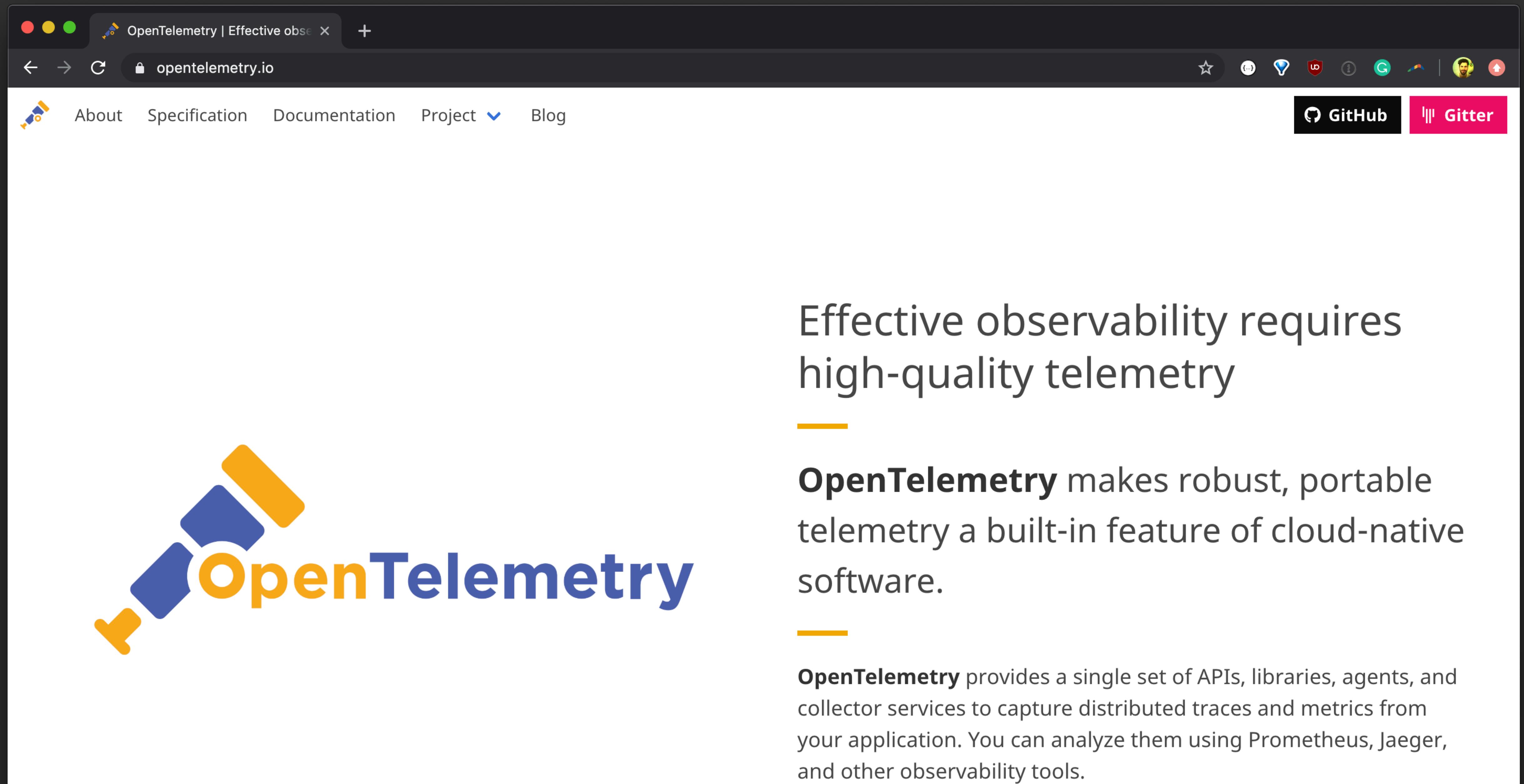
 OpenCensus is a set of libraries for various languages that allow you to collect application metrics and distributed traces, then transfer the data to a backend of your choice in real time. This data can be analyzed by developers and admins to understand the health of the application and debug problems.

[OVERVIEW](#) | [QUICKSTART](#)

Navigation sidebar (left side of the main content area):

- Introduction
- > Quickstart
- > Tracing
- > Stats/Metrics
- > Tags
- > Exporters
- > zPages
- > Service
- > Advanced Concepts
- > Guides
- Language Support
- > Integrations

opentelemetry.io



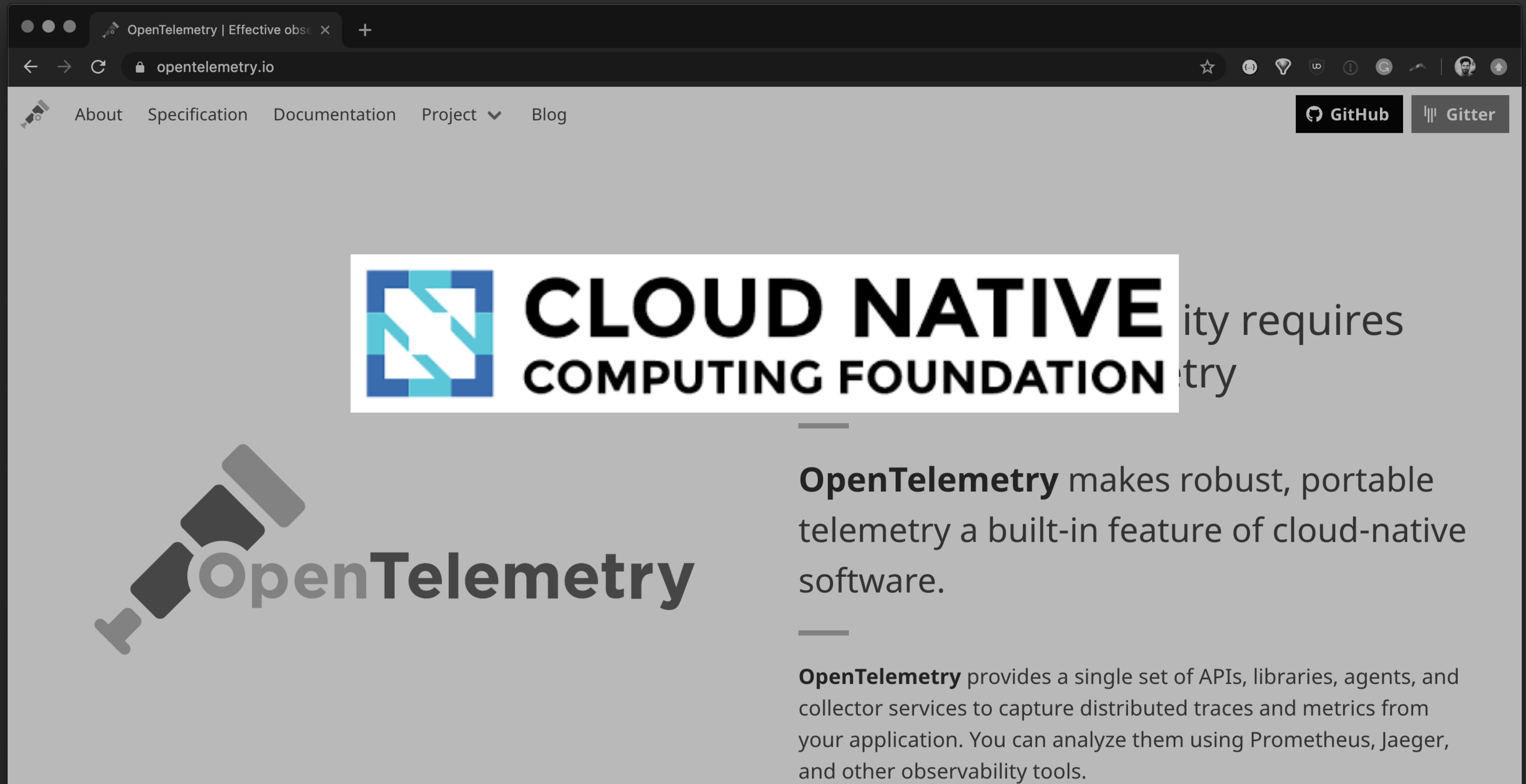
The screenshot shows the official OpenTelemetry website at opentelemetry.io. The page features a dark header with the site's logo and navigation links for About, Specification, Documentation, Project (with a dropdown menu), and Blog. On the right, there are GitHub and Gitter buttons. The main content area has a large heading "Effective observability requires high-quality telemetry" followed by three descriptive paragraphs about the project's goals and capabilities.

Effective observability requires high-quality telemetry

OpenTelemetry makes robust, portable telemetry a built-in feature of cloud-native software.

OpenTelemetry provides a single set of APIs, libraries, agents, and collector services to capture distributed traces and metrics from your application. You can analyze them using Prometheus, Jaeger, and other observability tools.

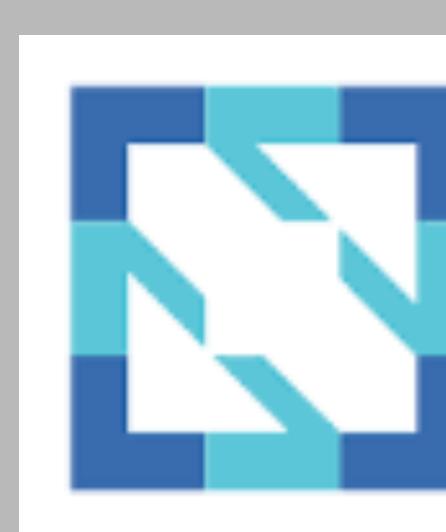
opentelemetry.io



The screenshot shows a web browser displaying the opentelemetry.io homepage. The address bar shows the URL "opentelemetry.io". The page features a navigation bar with links for "About", "Specification", "Documentation", "Project", and "Blog". On the right side of the header are buttons for "GitHub" and "Gitter". The main content area includes the Cloud Native Computing Foundation logo and a large "OpenTelemetry" logo with a stylized microphone icon. There are two sections of text describing the project's purpose and its API support.

OpenTelemetry | Effective observability

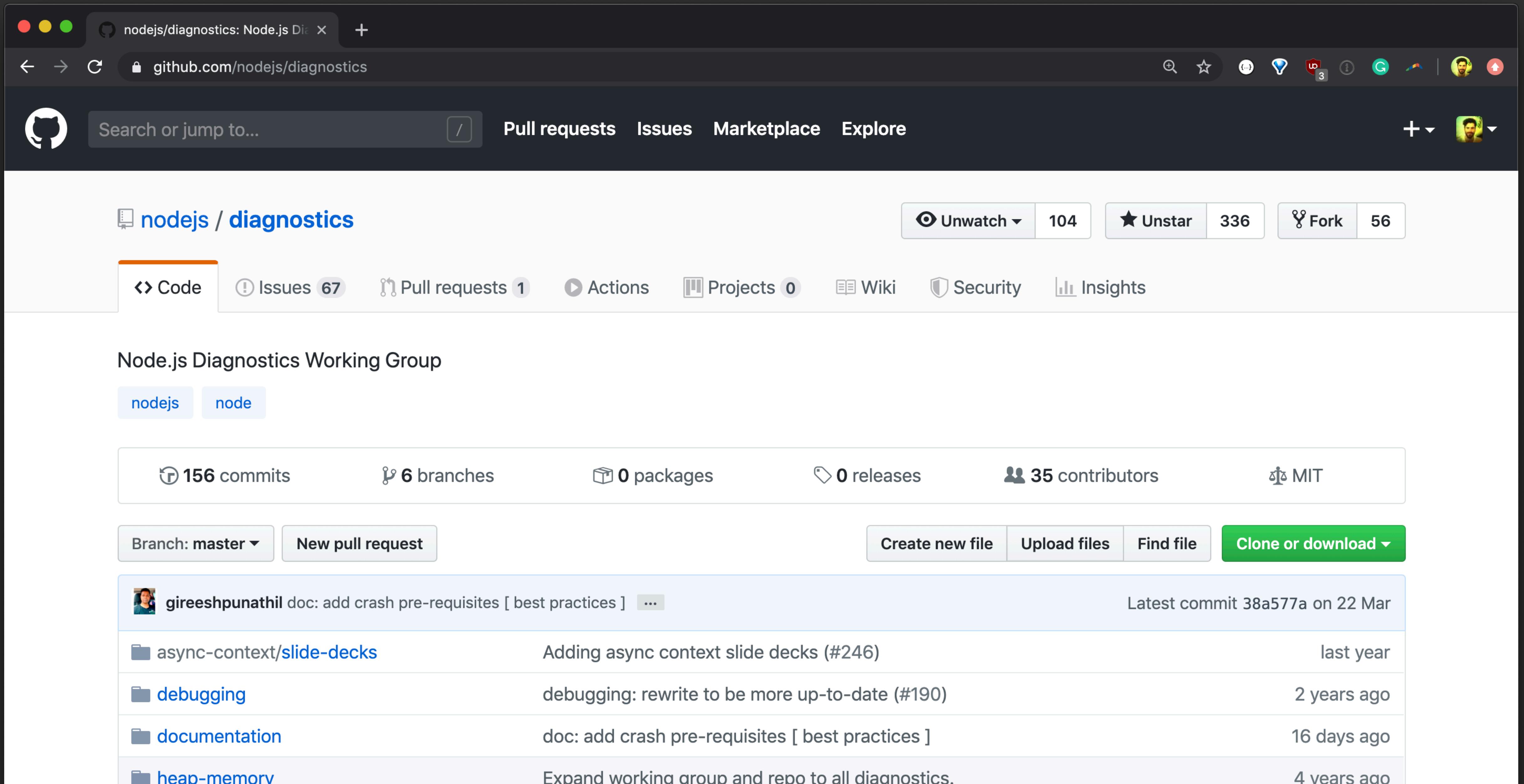
About Specification Documentation Project Blog

 CLOUD NATIVE COMPUTING FOUNDATION

OpenTelemetry makes robust, portable telemetry a built-in feature of cloud-native software.

OpenTelemetry provides a single set of APIs, libraries, agents, and collector services to capture distributed traces and metrics from your application. You can analyze them using Prometheus, Jaeger, and other observability tools.

github.com / nodejs / diagnostics



The screenshot shows the GitHub repository page for `nodejs/diagnostics`. The repository name is displayed at the top left, along with a search bar, a pull requests button, an issues button, a marketplace button, and an explore button. To the right are buttons for unwatching (104), unstarring (336), and forking (56). Below this, a navigation bar includes links for Code, Issues (67), Pull requests (1), Actions, Projects (0), Wiki, Security, and Insights. The main content area is titled "Node.js Diagnostics Working Group" and features two primary tabs: "nodejs" and "node". Key statistics are displayed: 156 commits, 6 branches, 0 packages, 0 releases, 35 contributors, and an MIT license. A "Clone or download" button is prominent. Below these stats, a dropdown for the branch "master" and a "New pull request" button are shown. A commit list is presented, starting with a recent commit by gireeshpunathil: "doc: add crash pre-requisites [best practices]" (commit 38a577a, 22 Mar). Other visible commits include "Adding async context slide decks (#246)" (last year), "debugging: rewrite to be more up-to-date (#190)" (2 years ago), "doc: add crash pre-requisites [best practices]" (16 days ago), and "Expand working group and repo to all diagnostics." (4 years ago).

nodejs/diagnostics: Node.js Dia X +

github.com/nodejs/diagnostics

Search or jump to... / Pull requests Issues Marketplace Explore

+ 

nodejs / diagnostics

Code Issues 67 Pull requests 1 Actions Projects 0 Wiki Security Insights

Unwatch 104 Unstar 336 Fork 56

Node.js Diagnostics Working Group

nodejs node

156 commits 6 branches 0 packages 0 releases 35 contributors MIT

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

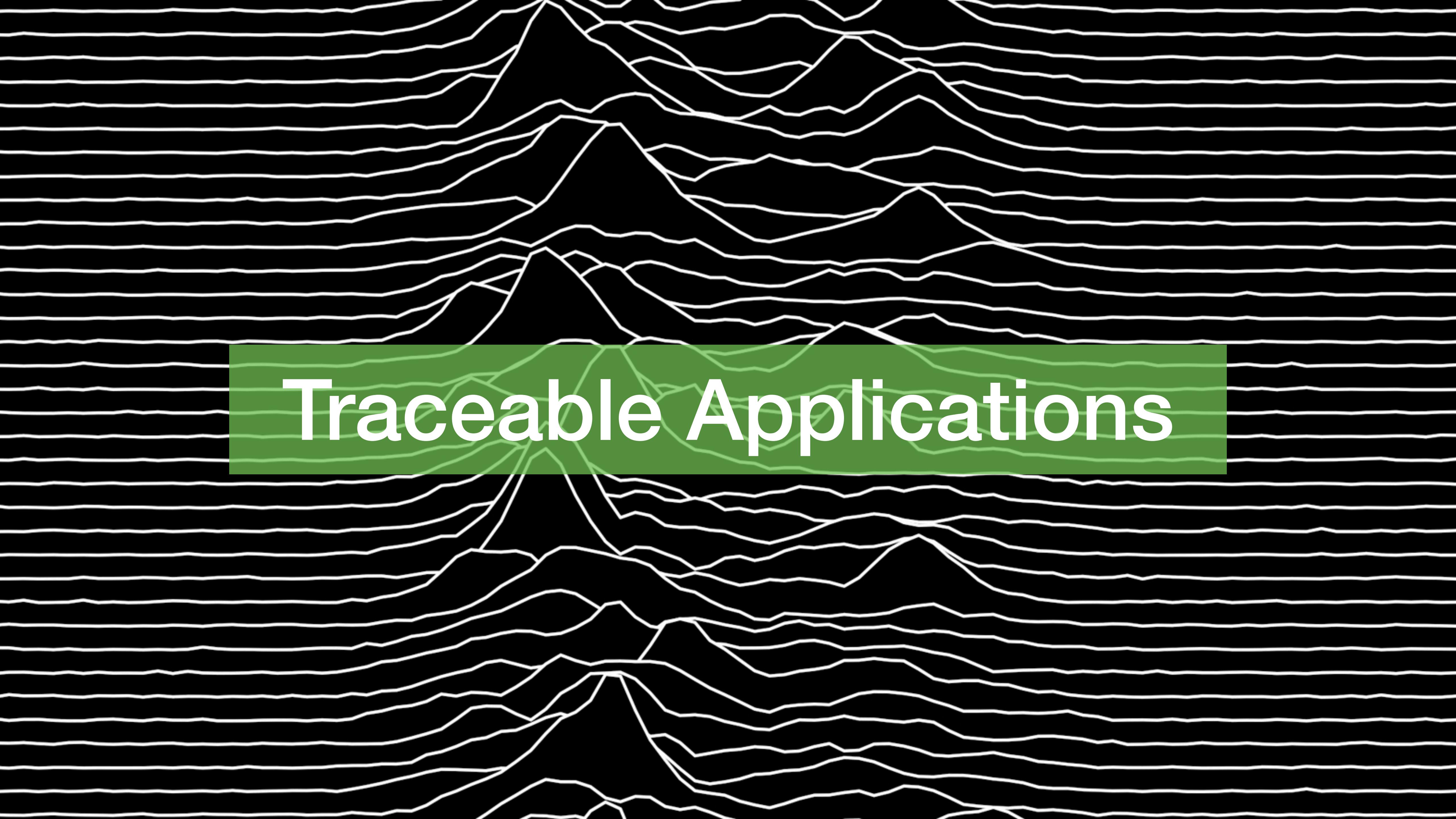
gireeshpunathil doc: add crash pre-requisites [best practices] ... Latest commit 38a577a on 22 Mar

async-context/slides-decks Adding async context slide decks (#246) last year

debugging debugging: rewrite to be more up-to-date (#190) 2 years ago

documentation doc: add crash pre-requisites [best practices] 16 days ago

heap-memory Expand working group and repo to all diagnostics. 4 years ago



Traceable Applications

Traceable Applications

- Challenges:
 - Thenables
 - Userland callback queue
 - Custom protocols, message queues etc
 - Storing all data

Dealing with Userland Callback Queues

- Tracer specific API
- Node.js Embedder API

```
class Client {
  constructor () {
    this.pool = new Pool({ size: 5 })
    this.queue = []
  }

  request (query, callback) {
    const socket = this.pool.getAvailableSocket()

    if (socket) {
      
      socket.send(query, (err, res) => {
        callback(err, res)

        if (this.queue.length > 0) {
          const [query, callback] = this.queue.shift()
          this.request(query, callback)
        }
      })
    } else {
      this.queue.push([query, callback])
    }
  }
}
```

**Queued callbacks
called in the context
of the previous
'socket.send()' call**

```
class Client {
  constructor () {
    this.pool = new Pool({ size: 5 })
    this.queue = []
  }

  request (query, callback) {
    const socket = this.pool.getAvailableSocket()

    if (socket) {
      socket.send(query, (err, res) => {
        callback(err, res)

        if (this.queue.length > 0) {
          const [query, callback] = this.queue.shift()
          this.request(query, callback)
        }
      })
    } else {
      callback = tracer.bindFunction(callback)
      this.queue.push([query, callback])
    }
  }
}
```

```
class Client {
  constructor () {
    this.pool = new Pool({ size: 5 })
    this.queue = []
  }

  request (query, callback) {
    const socket = this.pool.getAvailableSocket()

    if (socket) {
      socket.send(query, (err, res) => {
        callback(err, res)

        if (this.queue.length > 0) {
          const [query, callback] = this.queue.shift()
          this.request(query, callback)
        }
      })
    } else {
      callback = tracer.bindFunction(callback)
      this.queue.push([query, callback])
    }
  }
}
```

```
Tracer.prototype.bindFunction = function (fn) {
  const tracer = this
  const span = tracer.currentSpan

  return function wrapped () {
    const origSpan = tracer.currentSpan
    tracer.currentSpan = span

    fn.apply(this, arguments)

    tracer.currentSpan = origSpan
  }
}
```

```
Tracer.prototype.bindCallback = function (fn) {
    const tracer = this

    return function wrapper (...args) {
        const index = args.length - 1
        const callback = args[index]
        args[index] = tracer.bindFunction(callback)

        return fn.apply(null, args)
    }
}
```

```
const { AsyncResource } = require('async_hooks')

class DBQuery extends AsyncResource {
  constructor (client, query, callback) {
    super('DBQuery')
    this.client = client
    this.query = query
    this.callback = callback
  }

  execute () {
    this.client.request(this.query, (err, res) => {
      this.runInAsyncScope(this.callback, null, err, res)
    })
  }
}
```

```
const { AsyncResource } = require('async_hooks')

class DBQuery extends AsyncResource {
  constructor (client, query, callback) {
    super('DBQuery')
    this.client = client
    this.query = query
    this.callback = callback
  }

  execute () {
    this.client.request(this.query, (err, res) => {
      this.runInAsyncScope(this.callback, null, err, res)
    })
  }
}
```

```
const { AsyncResource } = require('async_hooks')

class DBQuery extends AsyncResource {
  constructor (client, query, callback) {
    super('DBQuery')
    this.client = client
    this.query = query
    this.callback = callback
  }

  execute () {
    this.client.request(this.query, (err, res) => {
      this.runInAsyncScope(this.callback, null, err, res)
    })
  }
}
```

```
const { AsyncResource } = require('async_hooks')

class DBQuery extends AsyncResource {
  constructor (client, query, callback) {
    super('DBQuery')
    this.client = client
    this.query = query
    this.callback = callback
  }

  execute () {
    this.client.request(this.query, (err, res) => {
      this.runInAsyncScope(this.callback, null, err, res)
    })
  }
}
```

```
const { AsyncResource } = require('async_hooks')

class DBQuery extends AsyncResource {
  constructor (client, query, callback) {
    super('DBQuery')
    this.client = client
    this.query = query
    this.callback = callback
  }

  execute () {
    this.client.request(this.query, (err, res) => {
      this.runInAsyncScope(this.callback, null, err, res)
    })
  }
}
```

```
const { AsyncResource } = require('async_hooks')

class DBQuery extends AsyncResource {
  constructor (client, query, callback) {
    super('DBQuery')
    this.client = client
    this.query = query
    this.callback = callback
  }

  execute () {
    this.client.request(this.query, (err, res) => {
      this.runInAsyncScope(this.callback, null, err, res)
    })
  }
}
```

```
const { AsyncResource } = require('async_hooks')

class DBQuery extends AsyncResource {
  constructor (client, query, callback) {
    super('DBQuery')
    this.client = client
    this.query = query
    this.callback = callback
  }

  execute () {
    this.client.request(this.query, (err, res) => {
      this.runInAsyncScope(this.callback, null, err, res)
    })
  }
}
```

```
class Client {
  constructor () {
    this.pool = new Pool({ size: 5 })
    this.queue = []
  }

  request (query, callback) {
    const socket = this.pool.getAvailableSocket()

    if (socket) {
      socket.send(query, (err, res) => {
        callback(err, res)

        if (this.queue.length > 0) {
          const resource = this.queue.shift()
          resource.execute()
        }
      })
    } else {
      this.queue.push(new DBQuery(this, query, callback))
    }
  }
}
```

```
class Client {
  constructor () {
    this.pool = new Pool({ size: 5 })
    this.queue = []
  }

  request (query, callback) {
    const socket = this.pool.getAvailableSocket()

    if (socket) {
      socket.send(query, (err, res) => {
        callback(err, res)

        if (this.queue.length > 0) {
          const resource = this.queue.shift()
          resource.execute()
        }
      })
    } else {
      this.queue.push(new DBQuery(this, query, callback))
    }
  }
}
```

```
class Client {
  constructor () {
    this.pool = new Pool({ size: 5 })
    this.queue = []
  }

  request (query, callback) {
    const socket = this.pool.getAvailableSocket()

    if (socket) {
      socket.send(query, (err, res) => {
        callback(err, res)

        if (this.queue.length > 0) {
          const resource = this.queue.shift()
          resource.execute()
        }
      })
    } else {
      this.queue.push(new DBQuery(this, query, callback))
    }
  }
}
```

nodejs.org / api / async_hooks.html

The screenshot shows a web browser window with the URL `nodejs.org/api/async_hooks.html` in the address bar. The page content is the documentation for the `async_hooks` module.

Node.js sidebar:

- About these Docs
- Usage & Example
- Assertion Testing
- Async Hooks**
- Buffer
- C++ Addons
- C/C++ Addons with N-API
- Child Processes
- Cluster
- Command Line Options
- Console
- Crypto
- Debugger
- Deprecated APIs
- DNS
- Domain
- ECMAScript Modules

Async Hooks page content:

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');
```

Terminology

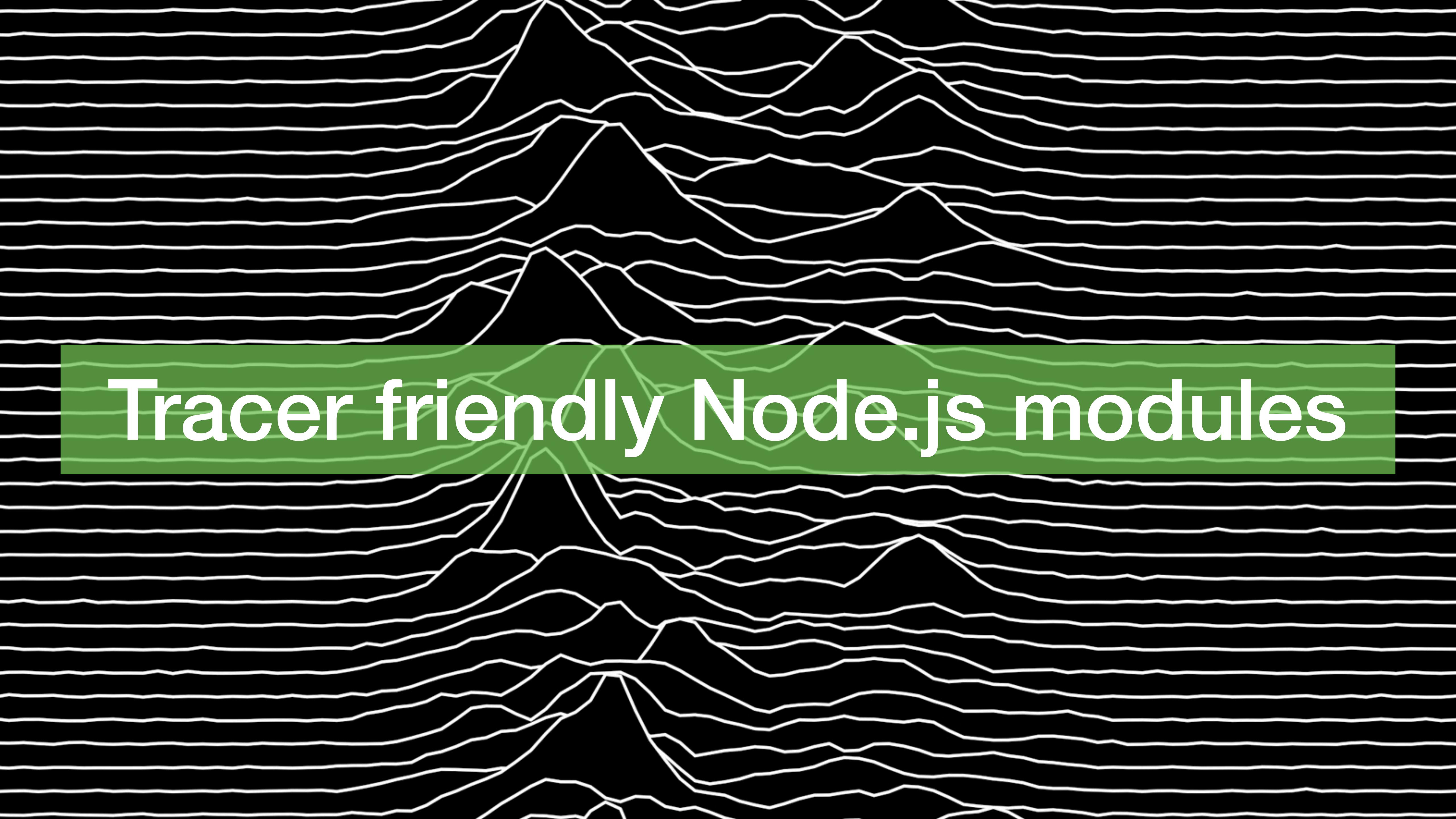
An asynchronous resource represents an object with an associated callback. This callback may be called multiple times, for example, the `'connection'` event in `net.createServer()`, or just a single time like in `fs.open()`. A resource can also be closed before the callback is called. `AsyncHook` does not explicitly distinguish between these different cases but will represent them as the abstract concept that is a resource.

If `Worker`s are used, each thread has an independent `async_hooks` interface, and each thread will use a new set of async IDs.

Public API

Overview

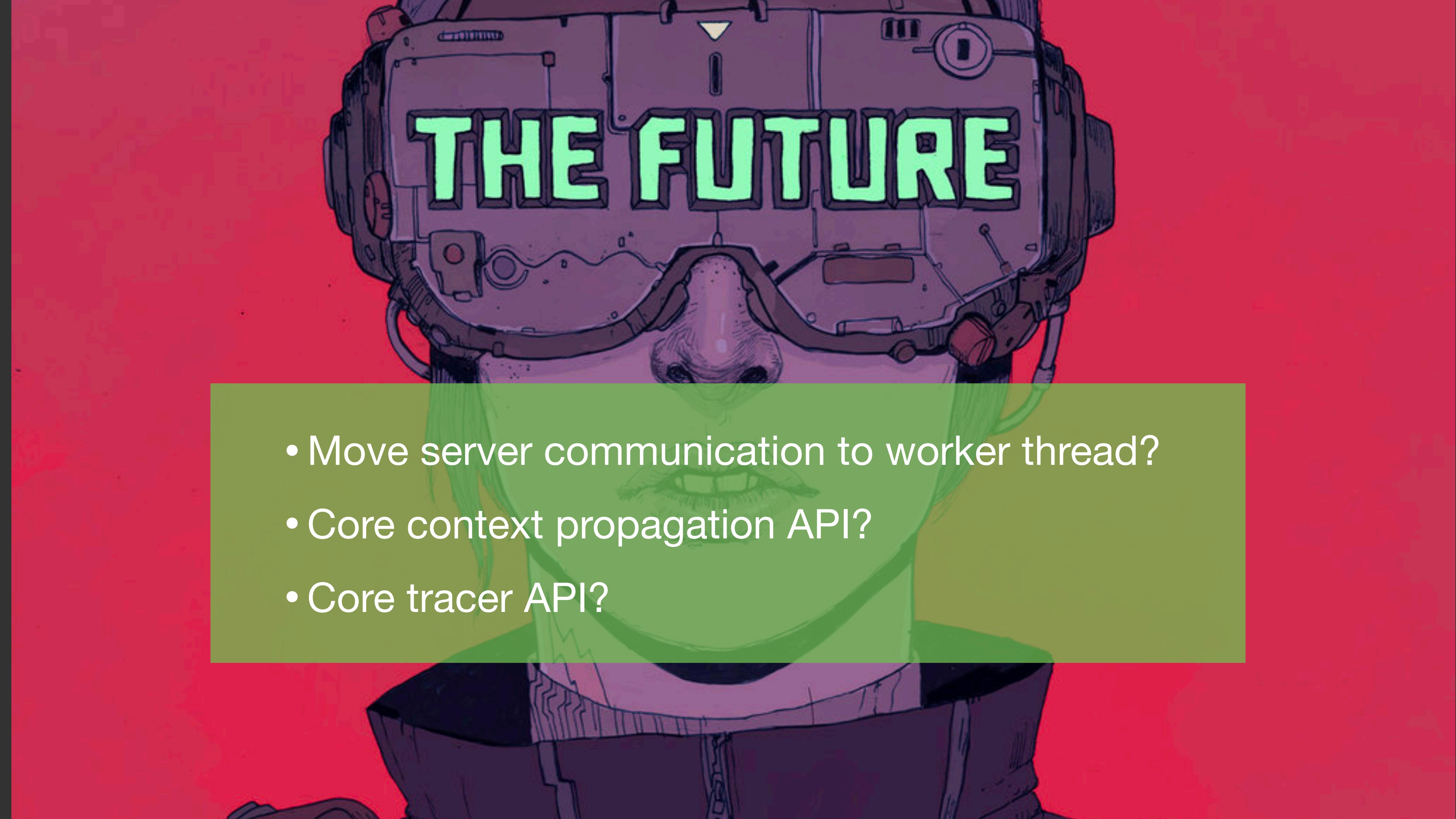
Following is a simple overview of the public API.



Tracer friendly Node.js modules

Tracer friendly Node.js modules

- Challenges:
 - Thenables 
 - Userland callback queue 
- Want:
 - Hooks or OpenTracing / OpenTelemetry compatibility



THE FUTURE

- Move server communication to worker thread?
- Core context propagation API?
- Core tracer API?

Благодаря



@wa7son

github.com/watson

