# Socket.IO & Open Academy

making fast human (data)-centric interactions on the webs

## What is Socket.IO?

- abstract protocol for event-based real-time applications on the web
- reference implementation in node.js
- uses the most advanced real-time data transfer technology available to a specific client under the hood

## How does “real-time” web programming work?

- Short Polling
  - Repeatedly ask server “Any new data?”
  - Server responds every time
- Long Polling
  - Ask server “Any new data?” and wait patiently for response
  - Server only responds when data is ready
- WebSockets
  - Open an instant 2-way communication channel between client and server

## Why use Socket.IO?

- Each browser supports a different subset of “real-time” approaches, so making a website for everyone to use is hard.
- Socket.IO abstracts the differences into an easy-to-use client & server model for the programmer.

## How?

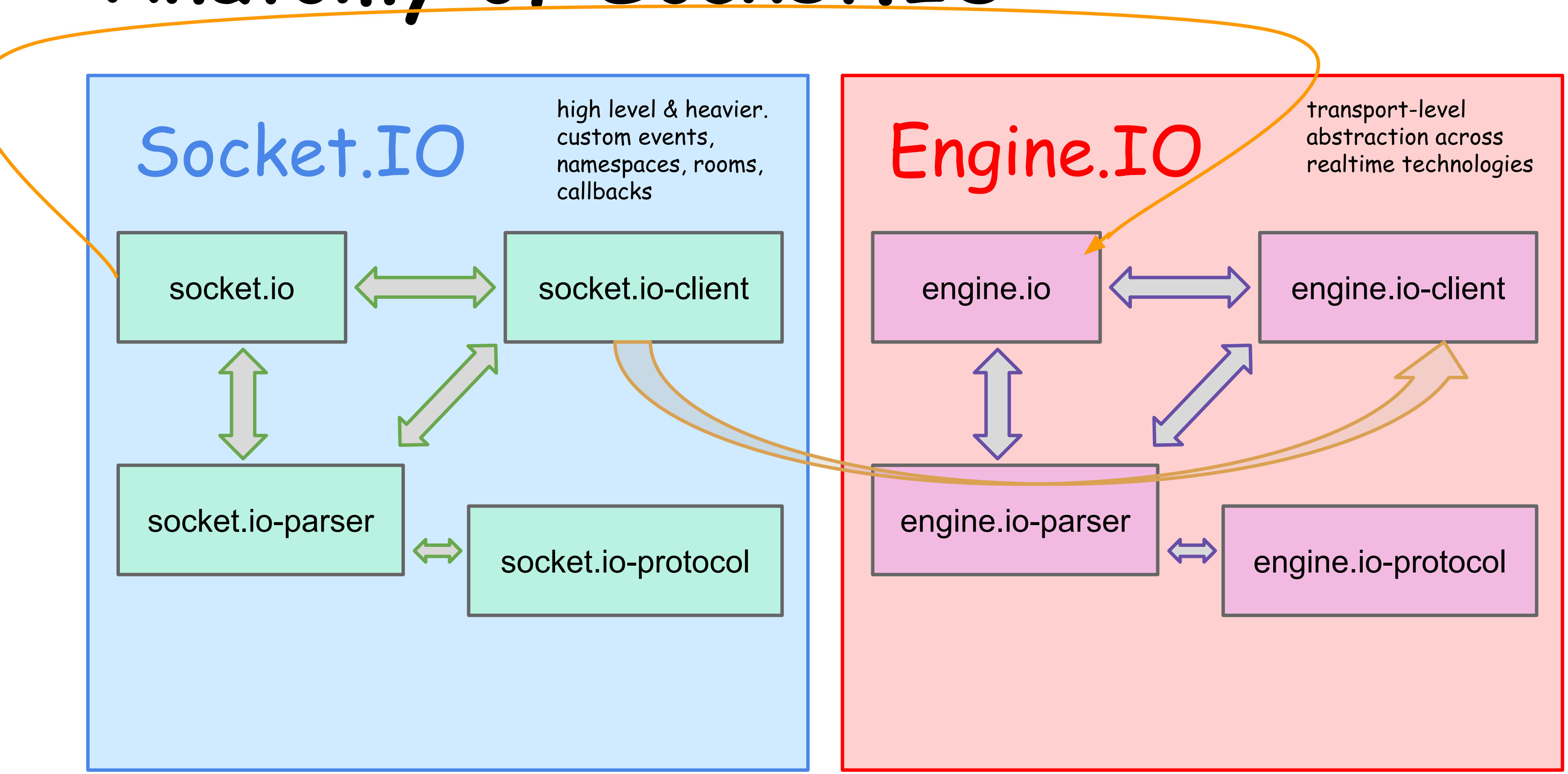
- **Server:**

```
io.on(‘connect’, function(socket) {
   socket.emit(‘hello’, ‘friend’);
});
```

- **Client:**

```
socket.on(‘hello’, function(name) {
   assert(name == ‘friend’);
   socket.emit(‘hey’, ‘you’);
});
```

# Anatomy of Socket.IO



## Time 4 us & our work

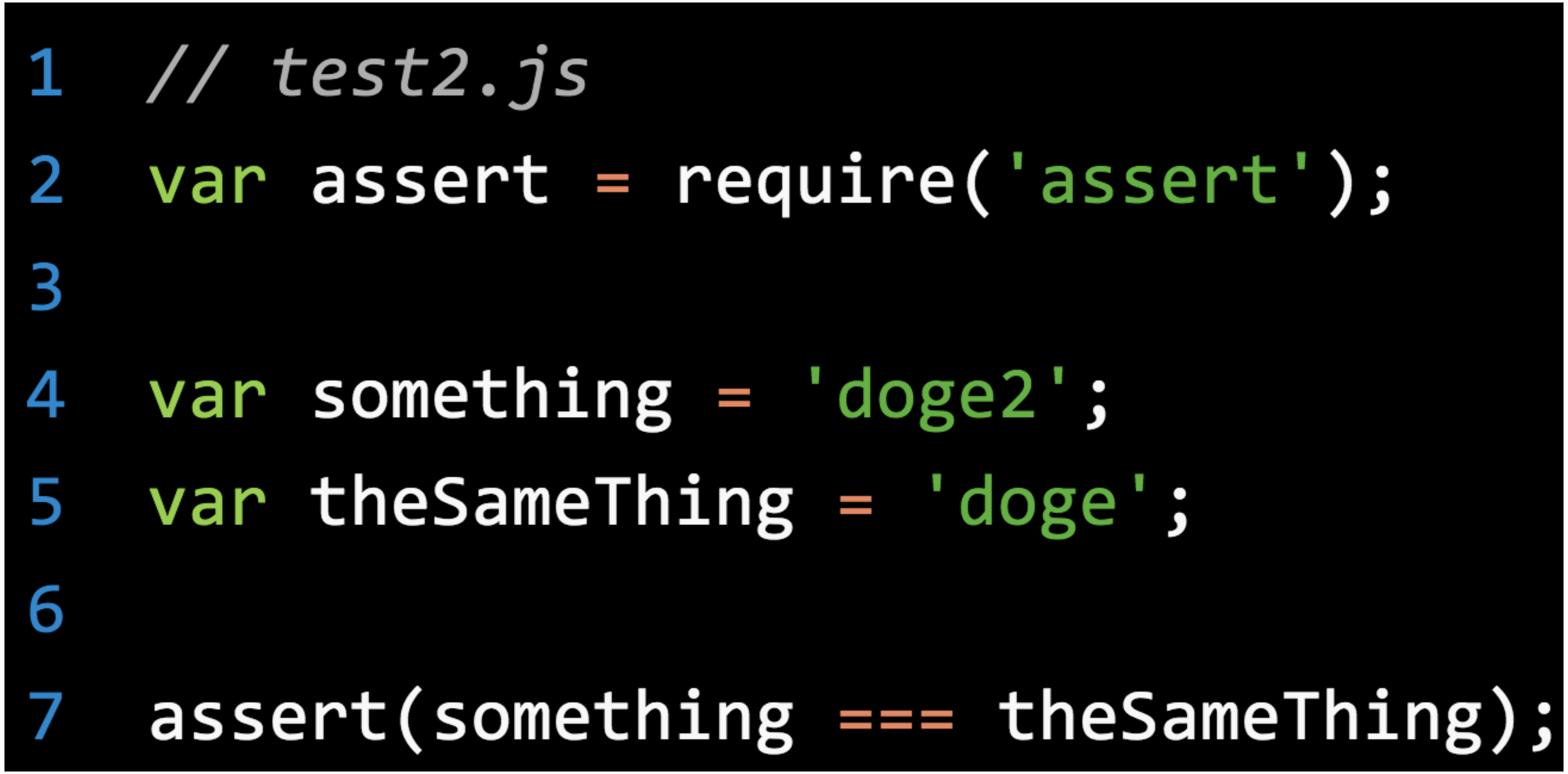
- Columbia's "Team IO"
  - Brian Shin
  - Adam Reis
  - Kevin Roark
- From facebook and donuts ——> New York and … bagels — all in real-time

## Goals

- Prepare the release of Socket.IO 1.0
  - Its been at 0.9.x for ~ 2 years.
  - (hint) 1.0 is almost complete!
  - https://github.com/LearnBoost/socket.io/
- Add binary support
- Fast, reliable testing to ensure that new versions work properly (server && client)
- Make some issues go away

## Adam’s Work

Let’s make assertions more useful

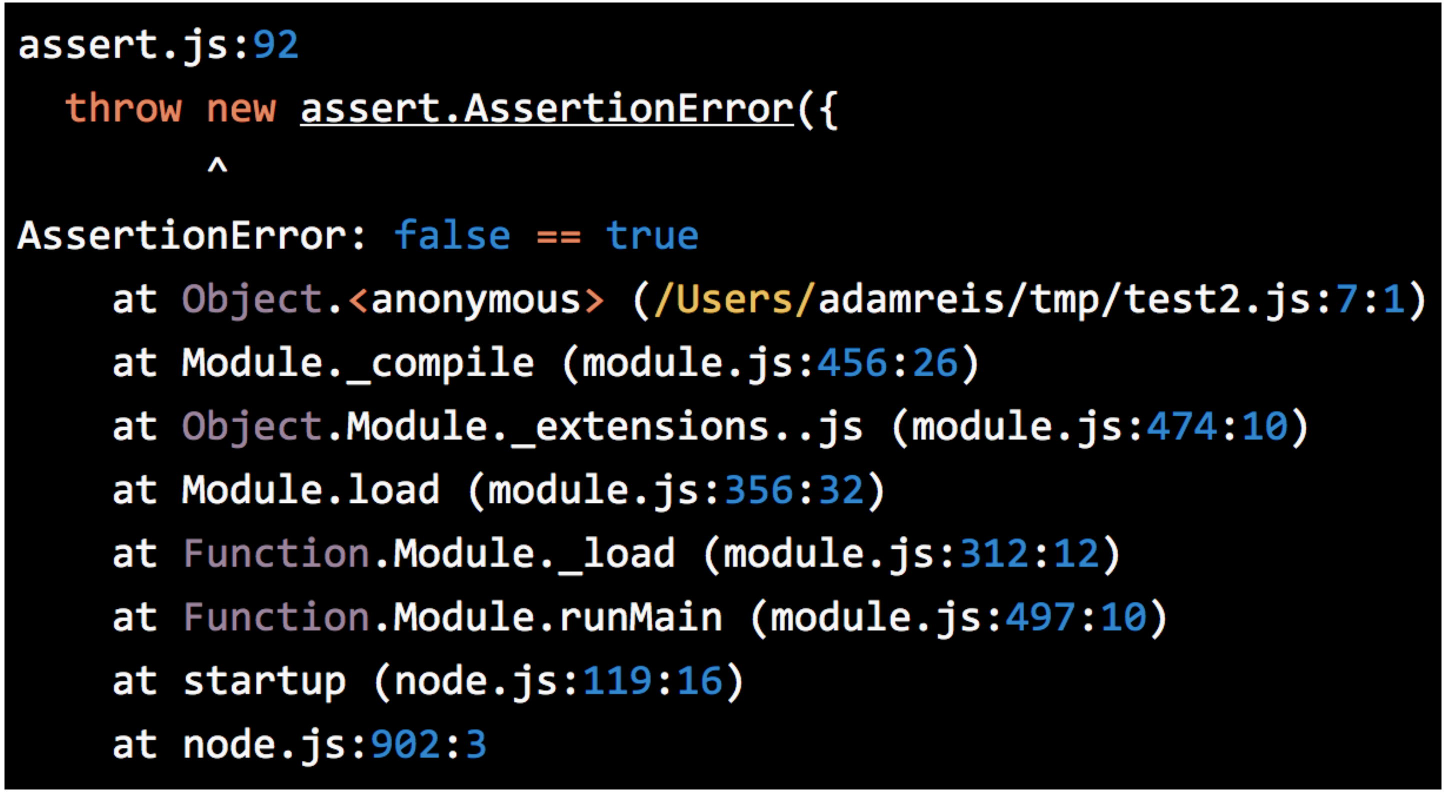


```
// test2.js
var assert = require('assert');

var something = 'doge2';
var theSameThing = 'doge';

assert(something === theSameThing);
```

## Bad



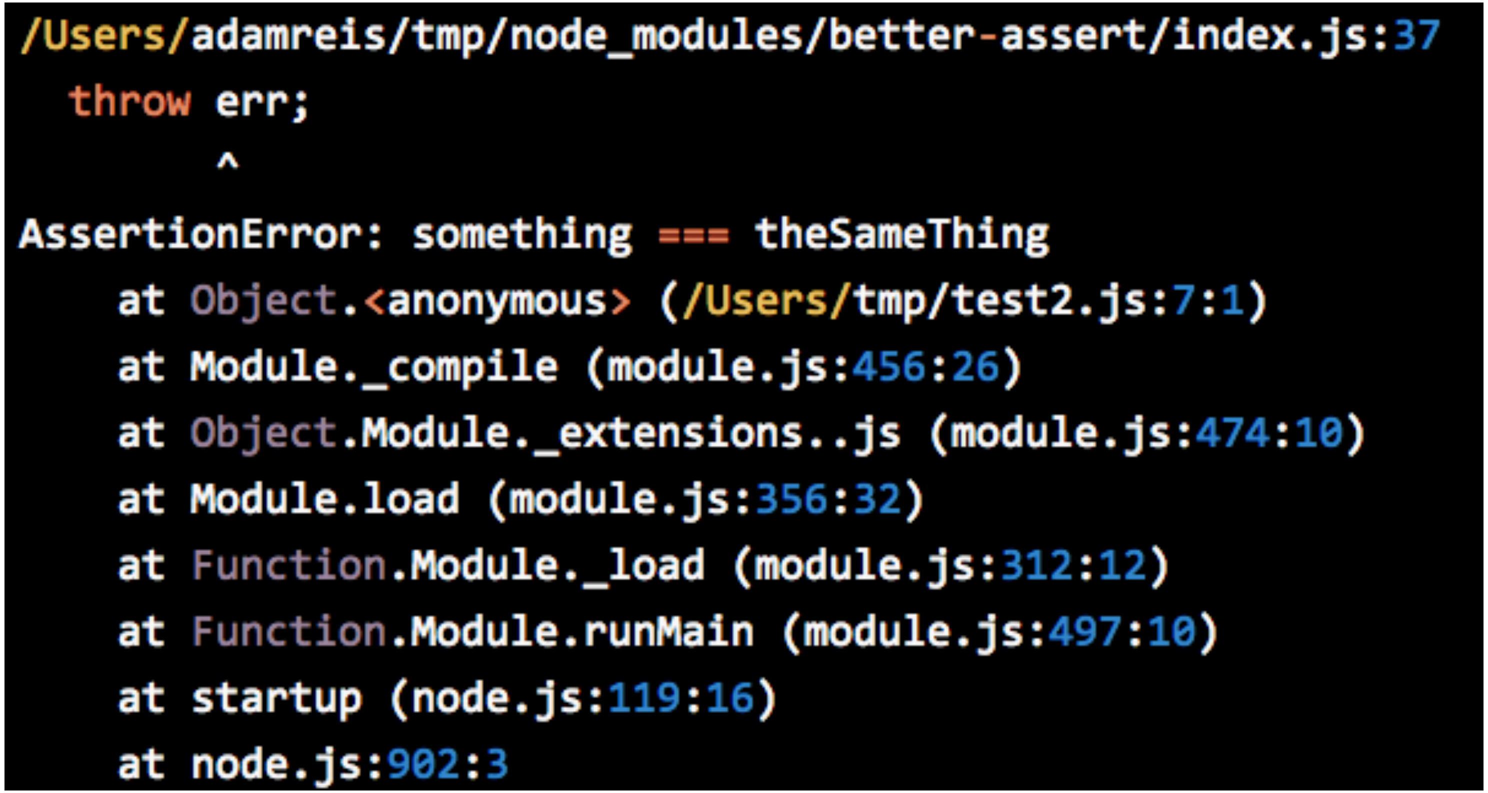
```
assert.js:92
  throw new assert.AssertionError({
        ^
AssertionError: false == true
    at Object.<anonymous> (/Users/adamreis/tmp/test2.js:7:1)
    at Module._compile (module.js:456:26)
    at Object.Module._extensions..js (module.js:474:10)
    at Module.load (module.js:356:32)
    at Function.Module._load (module.js:312:12)
    at Function.Module.runMain (module.js:497:10)
    at startup (node.js:119:16)
    at node.js:902:3
```

THIS DISPLEASES ME

## Current Solution

v8 JavaScript engine stack trace API

## Better



```
/Users/adamreis/tmp/node_modules/better-assert/index.js:37
  throw err;
        ^
AssertionError: something === theSameThing
    at Object.<anonymous> (/Users/tmp/test2.js:7:1)
    at Module._compile (module.js:456:26)
    at Object.Module._extensions..js (module.js:474:10)
    at Module.load (module.js:356:32)
    at Function.Module._load (module.js:312:12)
    at Function.Module.runMain (module.js:497:10)
    at startup (node.js:119:16)
    at node.js:902:3
```

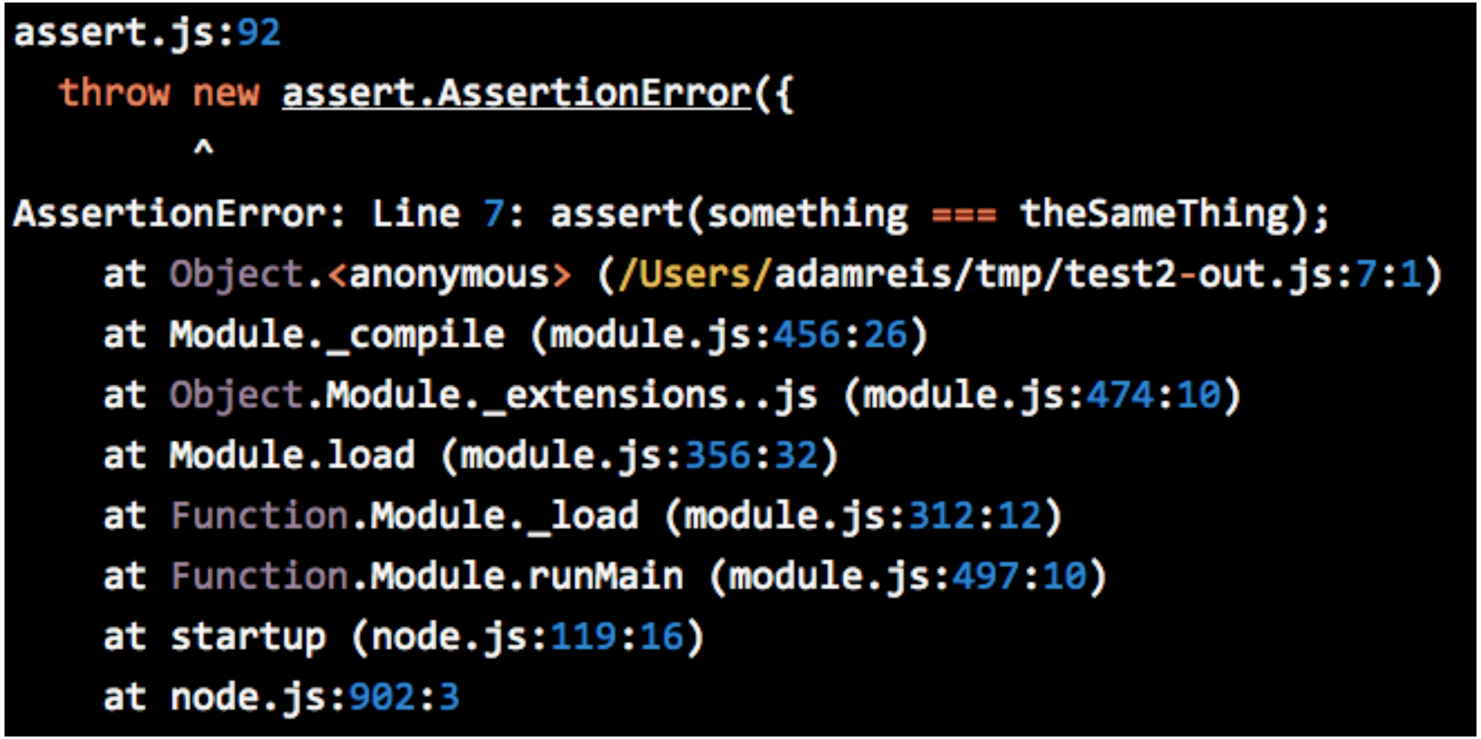
## Issue

Only works on browsers supporting v8

## My Solution: Super-Assert

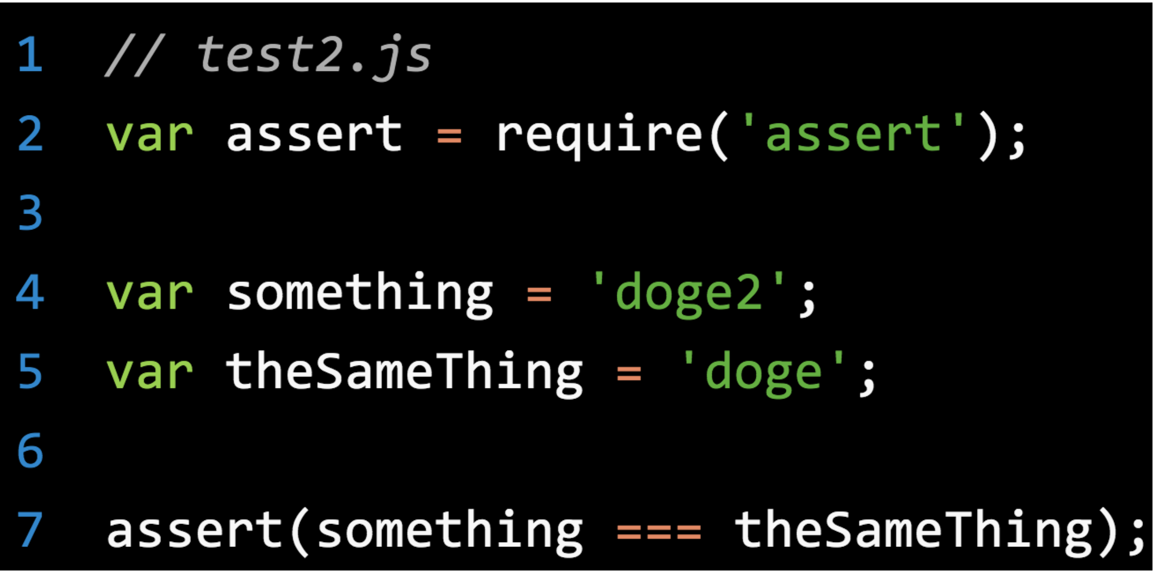
- Works on any browser

## Best



```
assert.js:92
  throw new assert.AssertionError({
        ^
AssertionError: Line 7: assert(something === theSameThing);
    at Object.<anonymous> (/Users/adamreis/tmp/test2-out.js:7:1)
    at Module._compile (module.js:456:26)
    at Object.Module._extensions..js (module.js:474:10)
    at Module.load (module.js:356:32)
    at Function.Module._load (module.js:312:12)
    at Function.Module.runMain (module.js:497:10)
    at startup (node.js:119:16)
    at node.js:902:3
```

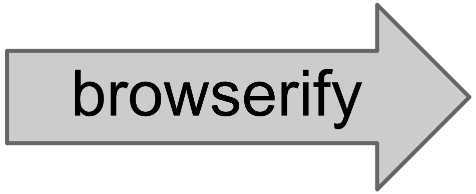
It even retains original line numbers through a browserify transform!



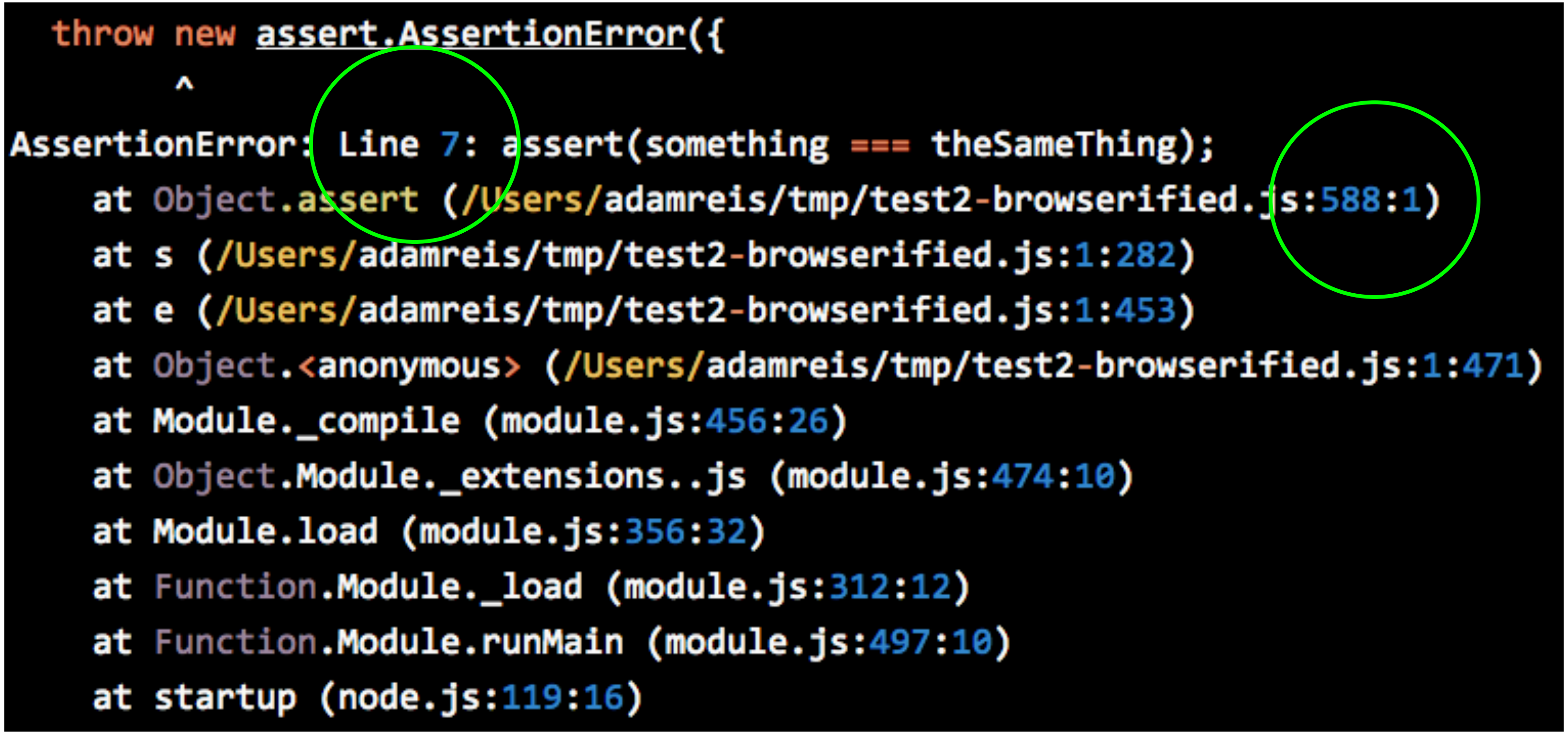
```
// test2.js
var assert = require('assert');

var something = 'doge2';
var theSameThing = 'doge';

assert(something === theSameThing);
```



## :)



```
  throw new assert.AssertionError({
        ^
AssertionError: Line 7: assert(something === theSameThing);
    at Object.assert (/Users/adamreis/tmp/test2-browserified.js:588:1)
    at s (/Users/adamreis/tmp/test2-browserified.js:1:282)
    at e (/Users/adamreis/tmp/test2-browserified.js:1:453)
    at Object.<anonymous> (/Users/adamreis/tmp/test2-browserified.js:1:471)
    at Module._compile (module.js:456:26)
    at Object.Module._extensions..js (module.js:474:10)
    at Module.load (module.js:356:32)
    at Function.Module._load (module.js:312:12)
    at Function.Module.runMain (module.js:497:10)
    at startup (node.js:119:16)
```

HERO
HUNN

## Kevin's Woark (get it)

- Binary support at Socket.IO level
- a number of little bugaroos
- weplay.io
- socket.io-computer
- # … #

## Binary

- could previously emit events with any valid JSON
- can now emit events that also contain buffers, blobs, files, and arraybuffers — arbitrary binary data
    - think images, sounds, all the good things
- Socket.IO can do *anything*

# Binary II

- most work done on socket.io-parser and protocol
  - more complex callback and event based encoding & decoding objects —> socket.io-parser is completely different
  - acknowledgement functions and broadcasting made it extra complex
  - interesting deconstruction and reconstruction of deep JSON with binary
  - improved first implementation's speed >> two-fold
- backed by new engine.io binary support (base64 fallback is very nice)

# Binary III

Class based model ::: why? binary is more complex and asynchronous so two functions aren't good enough. *ALSO*: goodbye msgpack.

```
module.exports.encode = function(packet) { /* return string encoding */ }
module.exports.decode = function(str) { /* return event packet */ }
```

VS

```
exports.Encoder = Encoder; function Encoder() {};
Encoder.encode = function(packet) { /* return array of encodings */ }


exports.Decoder = Decoder; function Decoder() {}; /* emits ‘decoded’ later
Decoder.add = function(encoding) { /* called whenever encoding received */
}
Decoder.destroy = function() { /* clean up */ }
```

# Binary IV

- Recognizing and translating anything that might be binary
- harder than you want it to be
- you have to dig deep
- related npm module: has-binary-data
- this is recognizing (translating is relatively similar):

```
function hasBin(obj) {
  if (Buffer.isBuffer(obj) || obj instanceof ArrayBuffer || …) { return true }
  else if (Array.isArray(obj) { /* go through every item */ }
  else if (typeof obj == ‘object’) { /* go through every key */ }
  return false;
}
```

## Binary V

things for the user aren’t different — they can now just emit more types of data.

its crazy to see complex and weird things working on internet explorer 6.

## weplay

- An example of the new binary feature's potential
- a pure JavaScript "clone" of twitch plays pokemon
- Collaborated with Guillermo & Tony, working on backend and frontend about equally (and some weird phantom stress testers!!)
- http://weplay.io/

## Bugs & small additions

- We know Socket.IO has many issues on github
- Here are a few of the smaller things I contributed:
    - query string parameters in 1.0
    - tests tests tests
    - documentation, like migration to 1.0
    - stupid javascript things like fixing bad type coercion
    - reducing build size & tricking browserify

## socket.io-computer

- a collaborative web-based virtual machine running windows xp
- Similar stack to weplay (redis, express, and binary socket.io), but this time I wrote every part of it and it's turn based
- almost done
- http://emu.weplay.io/

## fun for me!!

- learning the ins and outs of socket.io has made it very easy to use from the user's perspective
- *"Special Magic" things*
- *hi fi snock uptown*
- makes you appreciate how awesome and cool the project is!!

## Brian’s Work - Testing

- New version of socket.io, many new bugs
- SocketIO is just an abstraction
- Several underlying transport layers
  - Websockets
  - Polling (XHR, JSONP)
  - Flash
- Different code paths for different browsers

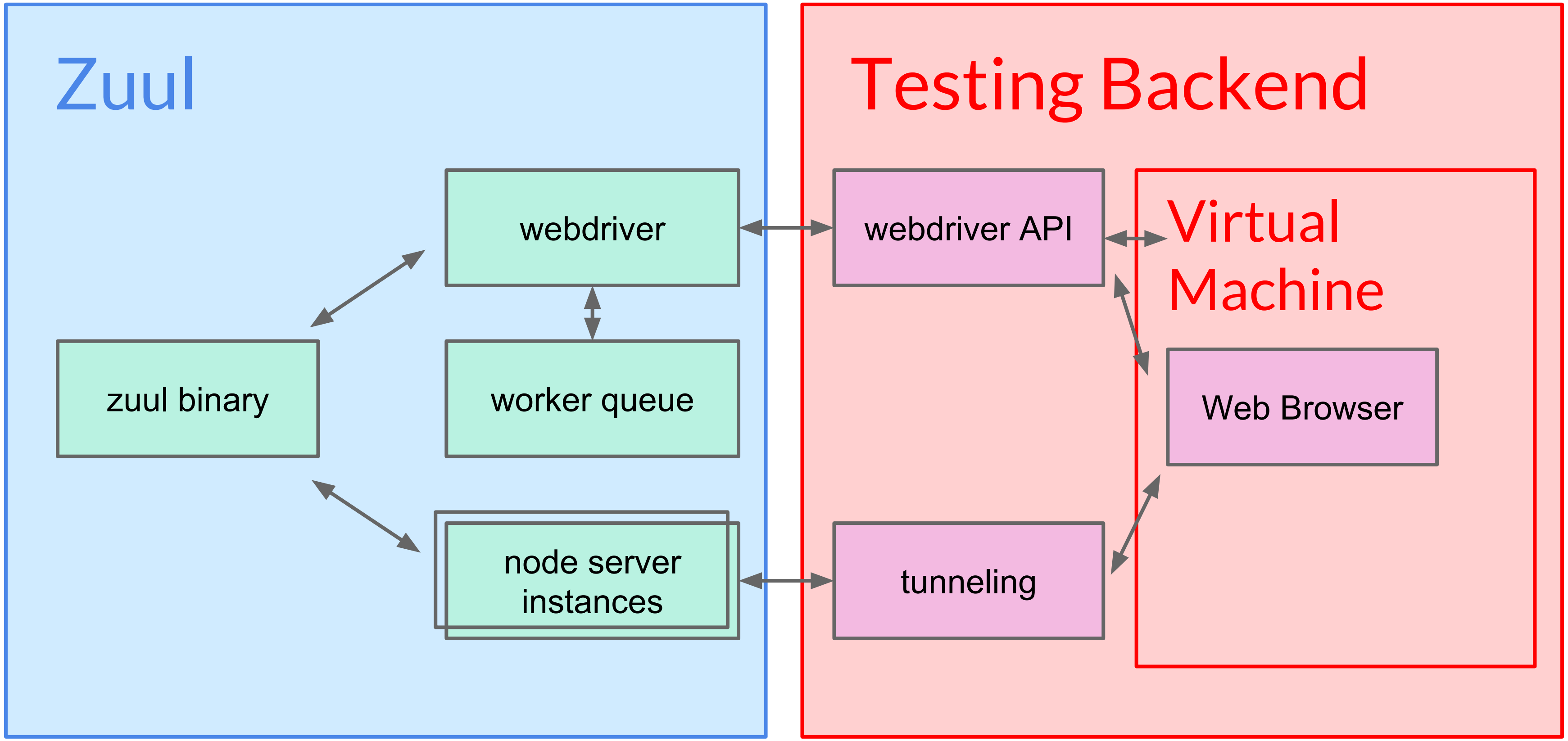
## Tests need to be...

- cross platform (including desktop, mobile, tablet)
- cross browser
- cross version
- very fast
    - many possible combinations of the above

## Zuul to the rescue

- Satisfies all of the above
- Runs tests concurrently for speed
- Abstraction over testing backends
- Abstraction over browser/version/OS selection
- Enables unit testing for browser APIs on different browser implementations

# Anatomy of Zuul

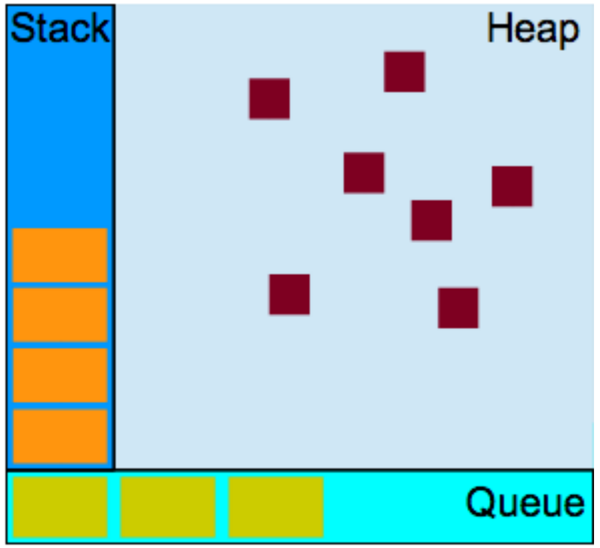


## Brian’s Work

- Better abstraction over non-local testing backends (other than executing shell commands)
- Added support for a new cloud backend - Browserstack
- Generic support for specifying browsers and platforms
- Reliable concurrency for remote execution

## Javascript Concurrency

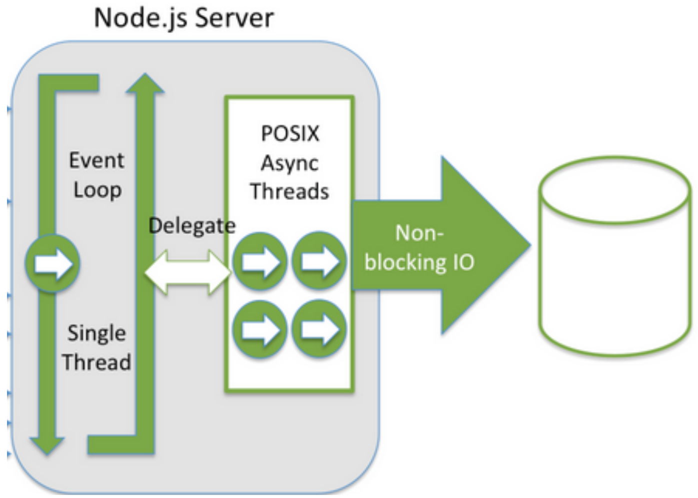
- Node.js and Browser Javascript operate using an event loop
- Node.js is single threaded
- Simple memory “sharing”



# Javascript Concurrency

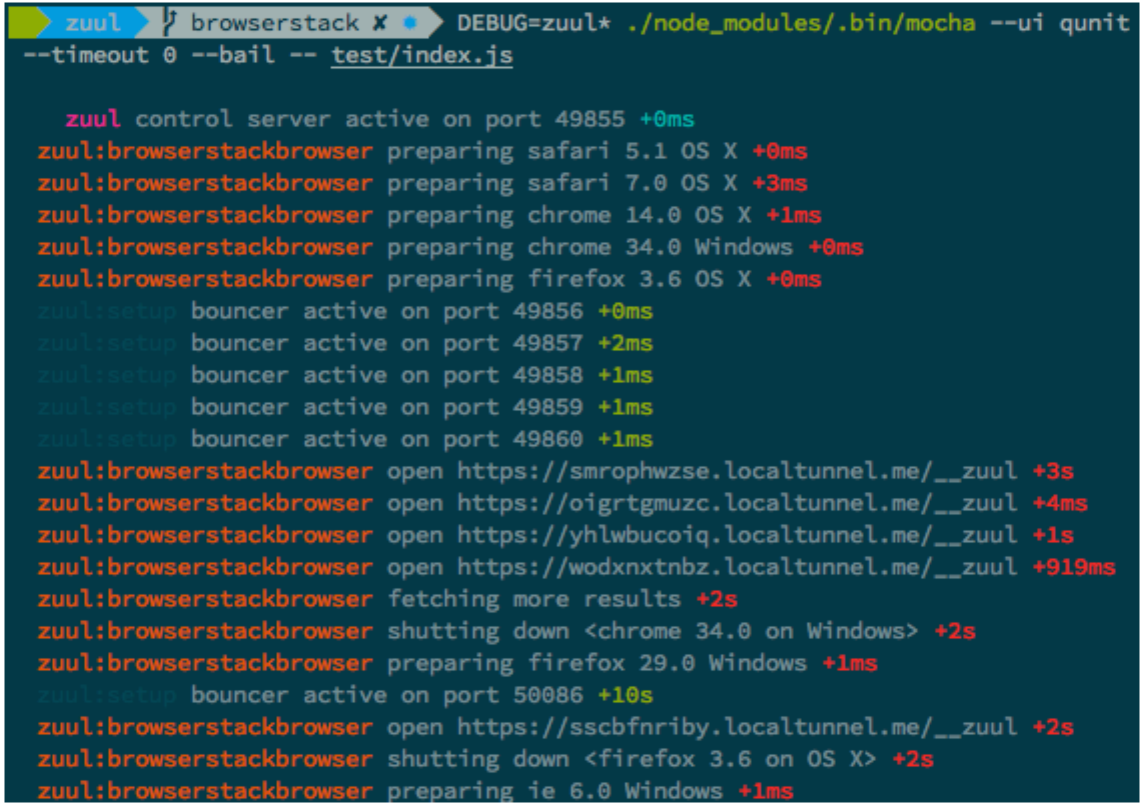
- I/O, however, happens in a separate thread
- Cannot use a `while` loop, must use `setTimeout` and callbacks

```
setTimeout(poll, 3000)
```



## Async Problems

- WebDriver API end event doesn’t always free the available slots immediately
- Concurrent requests will sometimes fail
- Solution: Introduction of a worker queue and status polling
- Coordinate workers by blocking queue while polling



```
 zuul  browserstack ✗ ●  DEBUG=zuul* ./node_modules/.bin/mocha --ui qunit
--timeout 0 --bail -- test/index.js

    zuul control server active on port 49855 +0ms
  zuul:browserstackbrowser preparing safari 5.1 OS X +0ms
  zuul:browserstackbrowser preparing safari 7.0 OS X +3ms
  zuul:browserstackbrowser preparing chrome 14.0 OS X +1ms
  zuul:browserstackbrowser preparing chrome 34.0 Windows +0ms
  zuul:browserstackbrowser preparing firefox 3.6 OS X +0ms
  zuul:setup bouncer active on port 49856 +0ms
  zuul:setup bouncer active on port 49857 +2ms
  zuul:setup bouncer active on port 49858 +1ms
  zuul:setup bouncer active on port 49859 +1ms
  zuul:setup bouncer active on port 49860 +1ms
  zuul:browserstackbrowser open https://smrophwzse.localtunnel.me/__zuul +3s
  zuul:browserstackbrowser open https://oigrtgmuzc.localtunnel.me/__zuul +4ms
  zuul:browserstackbrowser open https://yhlwbucoiq.localtunnel.me/__zuul +1s
  zuul:browserstackbrowser open https://wodxnxtnbz.localtunnel.me/__zuul +919ms
  zuul:browserstackbrowser fetching more results +2s
  zuul:browserstackbrowser shutting down <chrome 34.0 on Windows> +2s
  zuul:browserstackbrowser preparing firefox 29.0 Windows +1ms
  zuul:setup bouncer active on port 50086 +10s
  zuul:browserstackbrowser open https://sscbfnriby.localtunnel.me/__zuul +2s
  zuul:browserstackbrowser shutting down <firefox 3.6 on OS X> +2s
  zuul:browserstackbrowser preparing ie 6.0 Windows +1ms
```

## Reflections

- contributing real work and being involved in this community has been really awesome and an amazing experience a++++
- doge is high motivator \\\\ empirejs
- sometimes to work on a  big project you only need to know a small part
- beyond spring 14