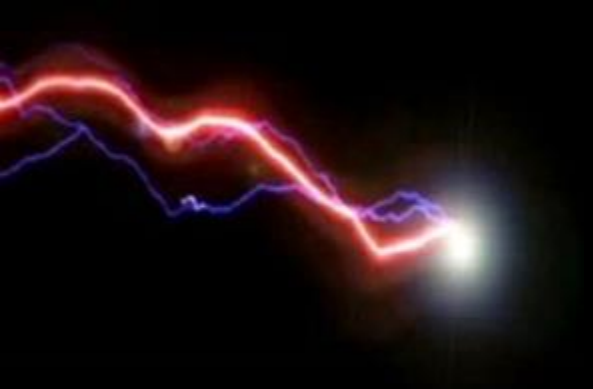
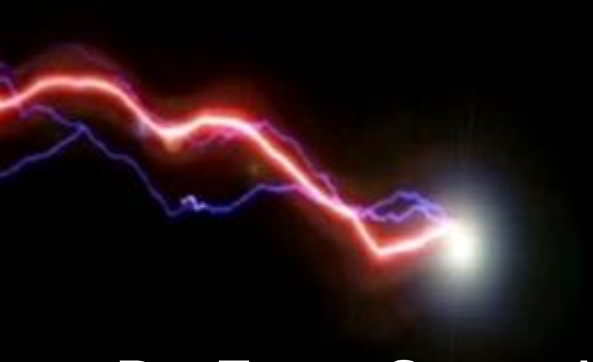


DON'T CROSS THE STREAMS

MATTHEW PODWYSOCKI
@MATTPODWYSOCKI
HTTP://GITHUB.COM/MATTPODWYSOCKI



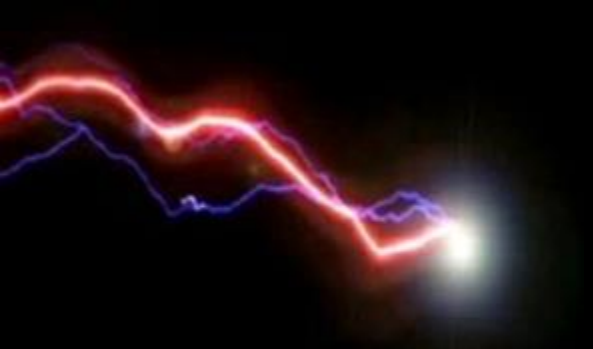
Dr. Egon Spengler: There's something very important I forgot to tell you.

Dr. Peter Venkman: What?

Dr. Egon Spengler: Don't cross the streams.

Dr. Peter Venkman: Why?

Dr. Egon Spengler: It would be bad.

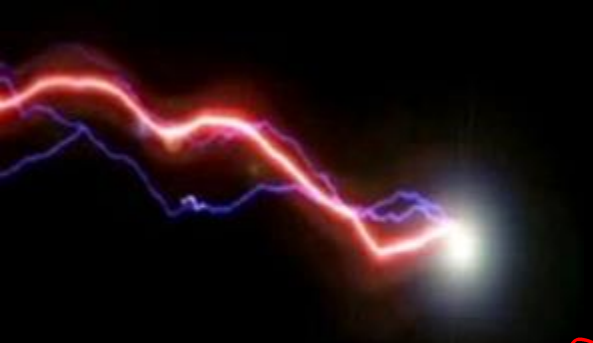


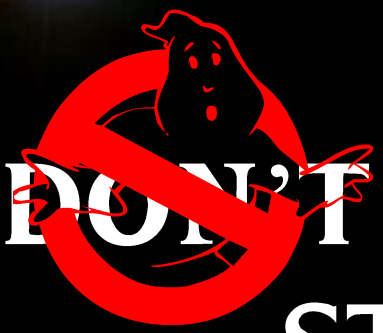
Dr. Peter Venkman: I'm fuzzy on the whole good/bad thing. What do you mean, "bad"?

Dr. Egon Spengler: Try to imagine all life as you know it stopping instantaneously and every molecule in your body exploding at the speed of light.

Dr Ray Stantz: Total protonic reversal.

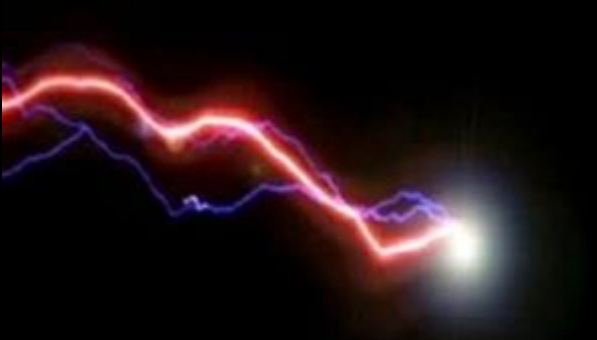
Dr. Peter Venkman: Right. That's bad. Okay. All right. Important safety tip. Thanks, Egon.





DON'T CROSS THE STREAMS

MATTHEW PODWYSOCKI
@MATTPODWYSOCKI
MATTHEWP@MICROSOFT.COM

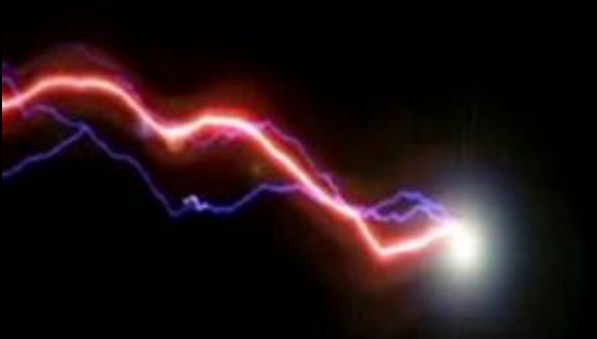


Dr. Egon Spengler: I have a radical idea. The door swings both ways, we could reverse the particle flow through the gate.

Dr. Peter Venkman: How?

Dr. Egon Spengler: We'll cross the streams.

Dr. Peter Venkman: 'Scuse me Egon? You said crossing the streams was bad!



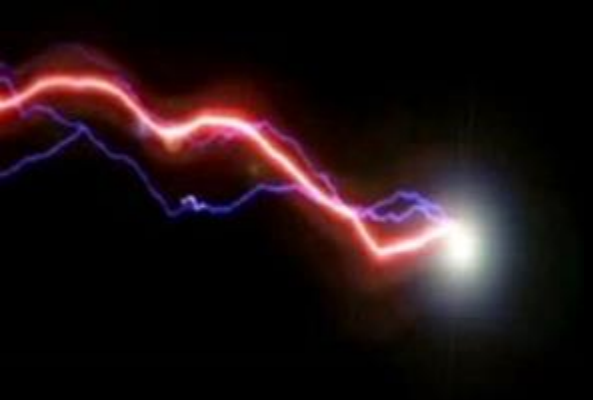
Dr Ray Stantz: Cross the streams...

Dr. Peter Venkman: You're gonna endanger us, you're gonna endanger our client - the nice lady, who paid us in advance, before she became a dog...

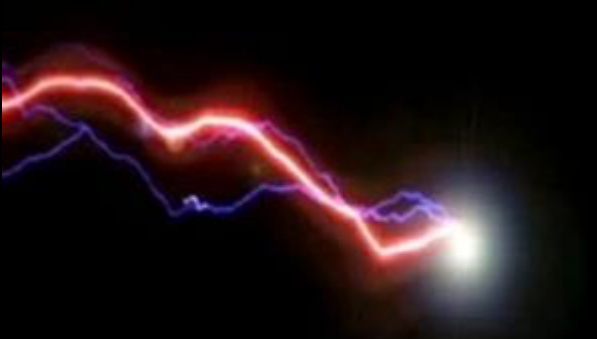
Dr. Egon Spengler: Not necessarily. There's definitely a *very slim* chance we'll survive.

Dr. Peter Venkman: I love this plan! I'm excited to be a part of it! LET'S DO IT!


MICROSOFT



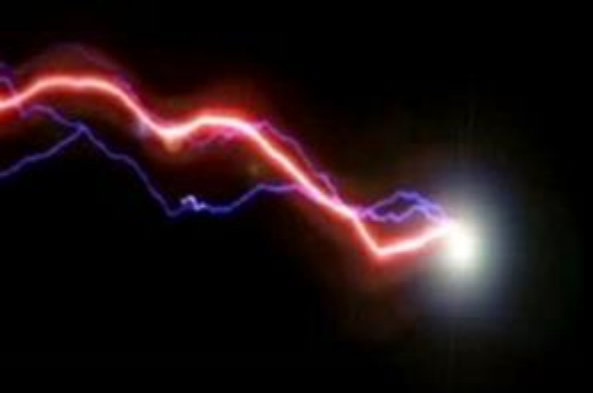

JS



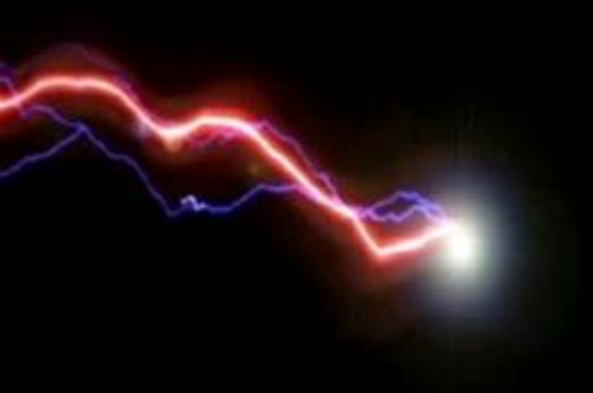
To put my strongest concerns into a nutshell:

1. We should have some ways of connecting programs like garden hose--screw in another segment when it becomes necessary to massage data in another way. This is the way of IO also.

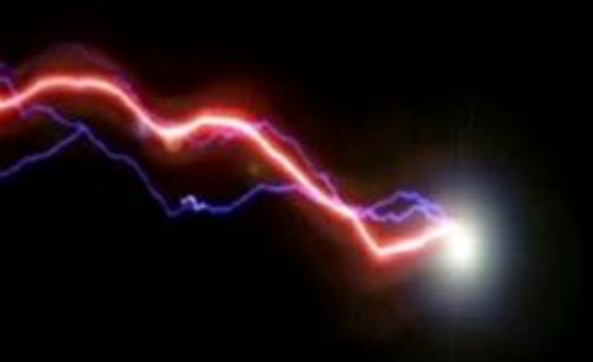
M. D. McIlroy
October 11, 1964



```
cat in.txt | tr '[A-Z]' '[a-z]' > out.txt
```

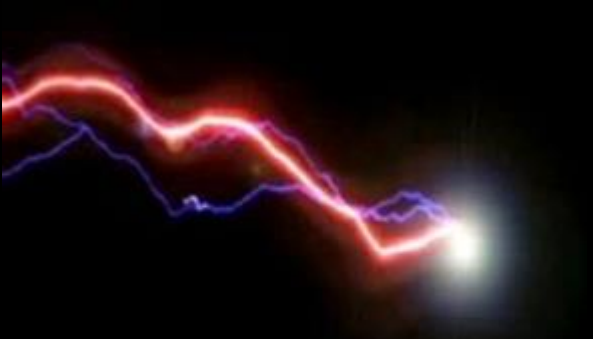


```
fs.createReadStream('in.txt')
  .pipe(quietStream())
  .pipe(fs.createWriteStream('out.txt'));
```



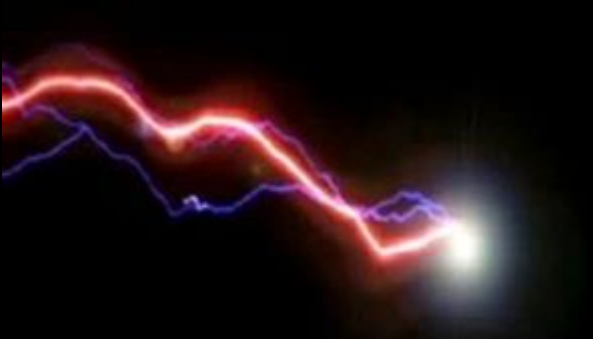
WHAT AND WHY?

- What
  - Abstraction of IO
  - Incremental data in time with back pressure
- Why
  - Improve Latency
  - Reduce memory footprint
  - Expand possibilities
  - Support Real-Time



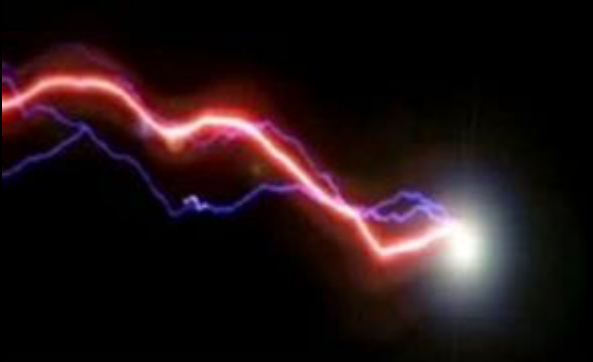
WHY USE THEM?

```
var http = require('http'),
    fs = require('fs');

http.createServer( function (req, res) {
    fs.readFile('file.txt', function (err, data) {
        if (err) {
            res.statusCode = 500;
            res.end(err.toString());
        }
        else res.end(data);
    });
});
```



WHY USE THEM?

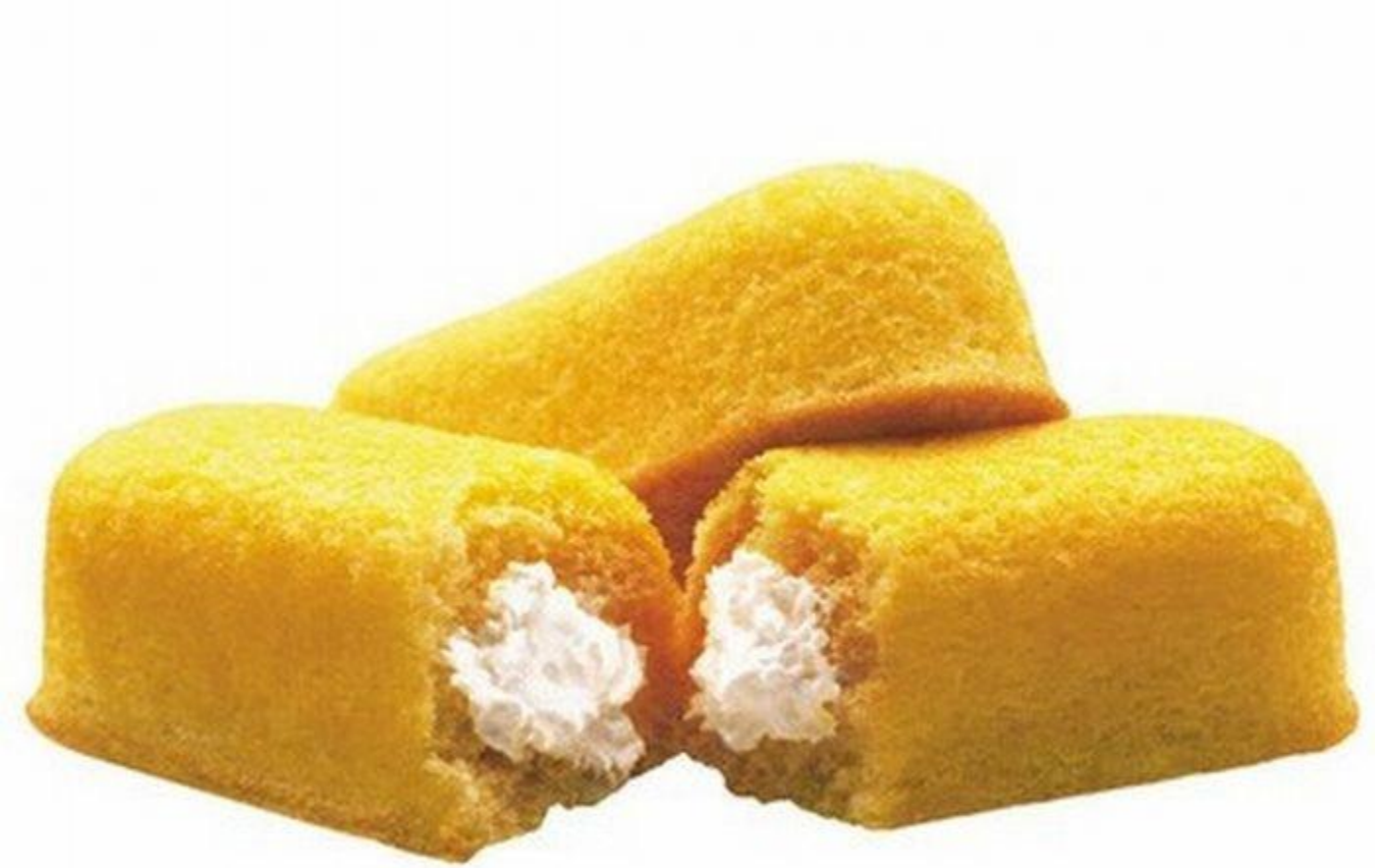
```
var http = require('http'),
    fs = require('fs');

http.createServer(function (req, res) {
    var s = fs.createReadStream('file.txt');
    s.on('error', function (err) {
        res.statusCode = 500;
        res.end(err.toString());
    };

    s.pipe(res);
});
```



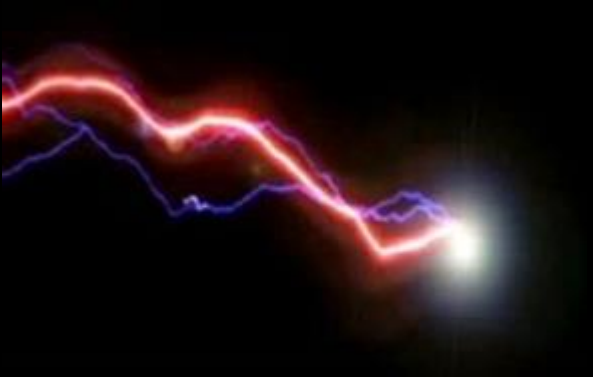
STREAMS...

- Subclass of EventEmitter
- Compose with pipe

```
var Stream = require('stream');

var s = new Stream();

...

s.pipe(process.stdout);
```



Well, let's say this Twinkie represents the normal amount of power in Node.js. Using the power of streams, it would be a Twinkie... thirty-five feet long, weighing approximately six hundred pounds.



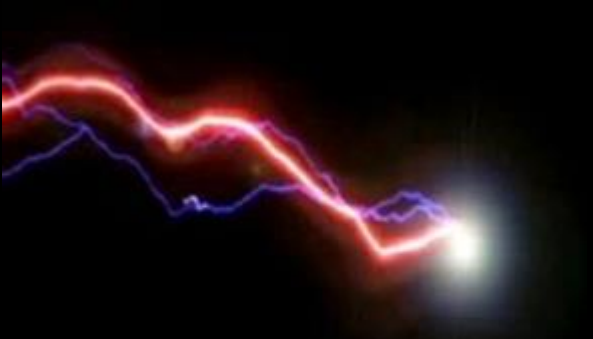
STREAMS...

READABLE

WRITEABLE

TRANSFORM

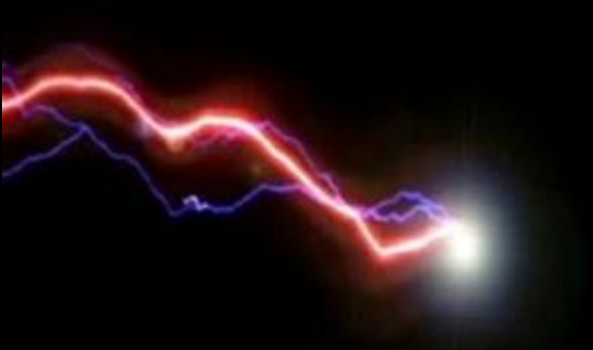
DUPLEX



READABLE

- Events: data, end, error, close
- Methods: pause, resume, destroy

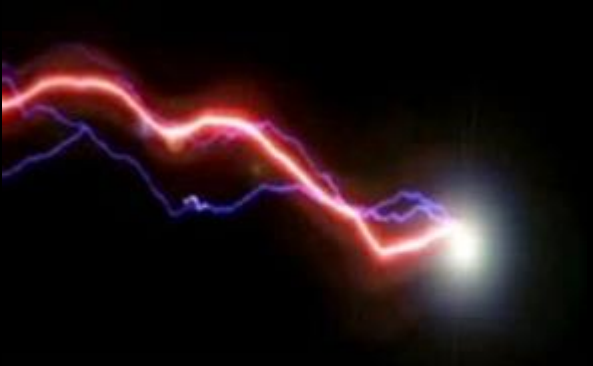
```
var s = new Stream();
s.readable = true;

var count = 0;
var id = setInterval(function () {
    s.emit('data', count + '\n');
    if (++count === 5) {
        s.emit('end');
        clearInterval(id);
    }
}, 1000);
```



WRITEABLE

- Events: drain, end, error, close
- Methods: write, end, destroy

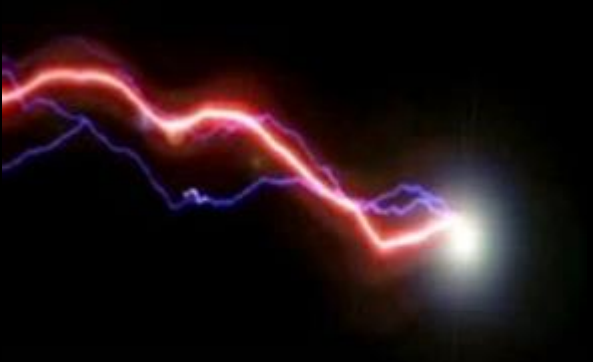
```
stream.writable = true;

s.write = function (data) { ... };

s.end = function (data) {
    if (arguments.length) s.write(data);
    this.destroy();
};

s.destroy = function () {
    this.writable = false;
};
```



BACK PRESSURE

- Ensure Readable streams don’t emit faster than Writeable streams can consume
- Drastically changing with Node > 0.8

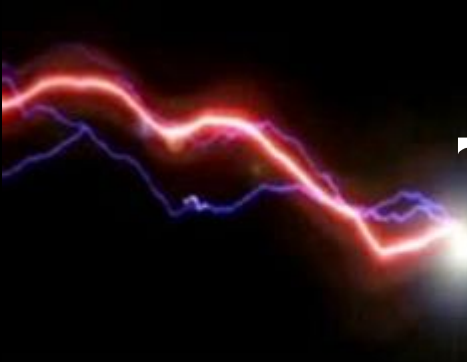
```
writer.write() === false   reader.pause()

writer.emit('drain')       reader.resume()
```



PIPE

Readable Stream can be piped to a writable stream while handling backpressure

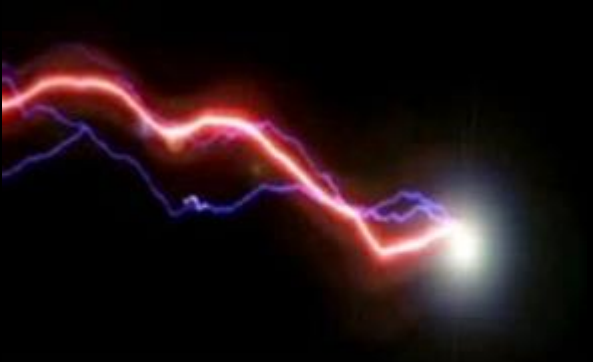
Readable Stream		Writable Stream
`emit('data', data)`		`write(data)`
`emit('end')`		`end()`
`pause()`		`write() === false`
`resume()`		`emit('drain')`
`emit('close')`		`destroy()`



TRANSFORM STREAMS

- Both readable and writable
- Transform input and produce result

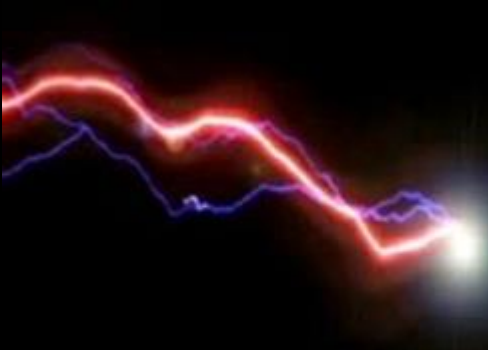
```
readable.pipe(through).pipe(writable)
```



DUPLEX STREAMS

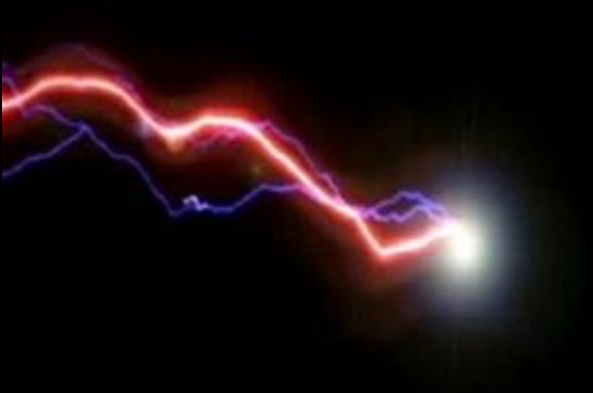
- Both readable and writable
- Both ends of the engage in a two-way interaction

```
stream1.pipe(stream2).pipe(stream1);
```



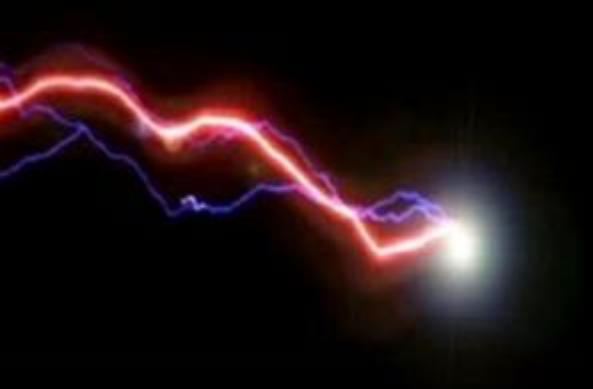
BATTERIES INCLUDED

- process.stdin, stdout, stderr
- net
- http
- fs
- child_process
- zlib



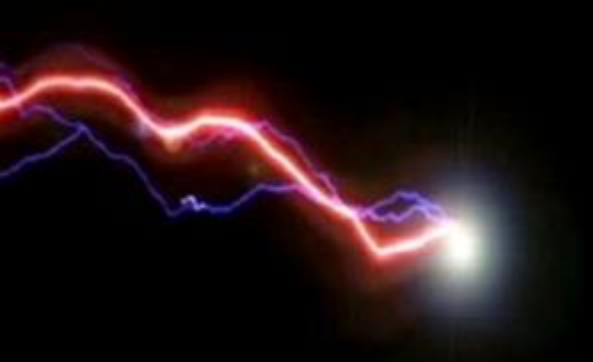
request, filed, JSONStream, mux-demux, shoe,
pause-stream, emit-stream, through,
scuttlebutt, tar, dnode, event-stream

WHO YOU GONNA CALL?



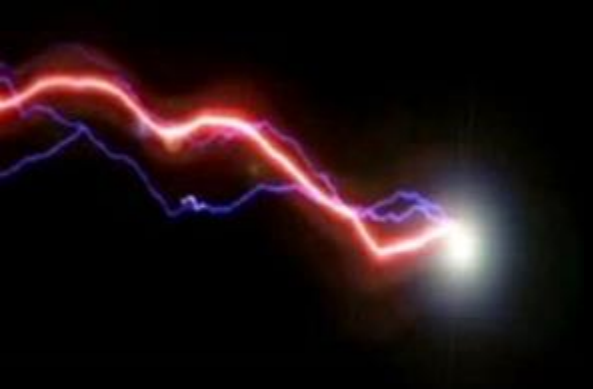
substack, dominictarr, maxogden, mikeal,
polotek, isaacs, raynos, fent, tootallnate

WE'RE READY TO BELIEVE YOU



ISSUES IN ≤ 0.8

- Problems in Readable Streams
  - Data eagerly fired whether ready or not
  - Implement pause/resume yourself
  - Might get data even if paused



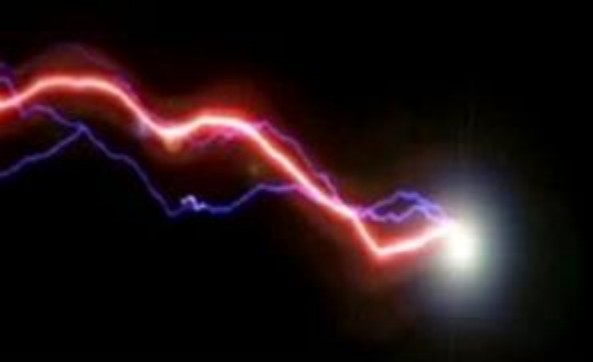
ISSUES IN $\leq$ 0.8

“Fire and brimstone coming down from the skies! Rivers and seas boiling!”

“Forty years of darkness! Earthquakes, volcanoes…”

“The dead rising from the grave!”

“Human sacrifice, dogs and cats living together… mass hysteria!”

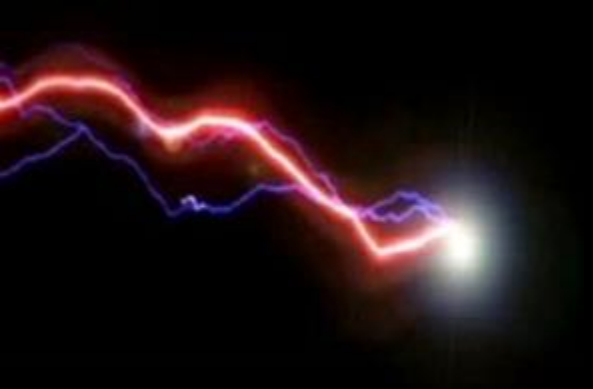


CHANGES IN 0.9+

- New Readable class
  - Eliminates pause/resume
  - Adds read method and readable event

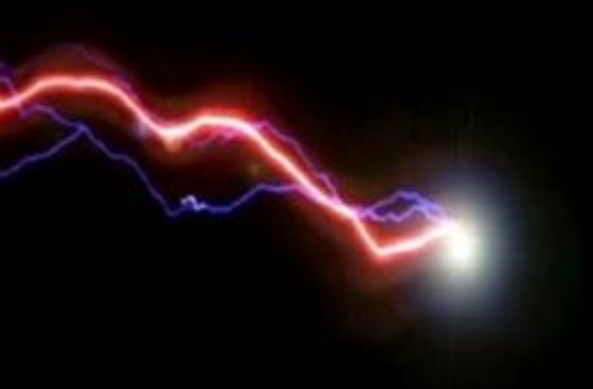
```
function flow() {
  var chunk;
  while ((chunk = r.read()) !== null) {
    process(chunk);
  }
  r.once('readable', flow);
}
flow();
```

https://github.com/isaacs/readable-stream



STREAM HANDBOOK

https://github.com/substack/stream-handbook



CROSS THE STREAMS

MATTHEW PODWYSOCKI
@MATTPODWYSOCKI
MATTHEWP@MICROSOFT.COM

CREDITS

- Proton Stream: http://current.com/technology/90461049_las-vegas-ghostbusters-proton-stream-test.htm
- Twinkie: http://www.pics-site.com/2011/01/27/a-twinkie-in-a-ct-scanner/