Streams, Pipes and Mega Pipes

Felix Geisendörfer

11.06.2011 (v1)

@felixge

Twitter / GitHub / IRC

Co-founder transloadit.com

Core Developer

&

Module Author

node-mysql

node-formidable

Streams

Image Resizing Server

```
var http = require('http');
var spawn = require('child_process').spawn;

http.createServer(function(req, res) {
  var params = req.url.split('/');
  var path = __dirname + '/' + params[1];
  var size = params[2];

  var convert = spawn('convert',  [path, '-resize', size, '-']);
  res.writeHead(200);
  convert.stdout.pipe(res);
}).listen(8080);
```

On Github

felixge/node-convert-example

"Streams are to time as arrays are to space."

-- Jed Schmidt @ JSConf.eu

Readable Streams

```
var fs = require('fs');
var stream = fs.createReadStream('/dev/random');

stream.on('data', function(buffer) {
  console.log(buffer);
});
```

```
$ node read.js
<Buffer 30 85 85 f1 33 f3 4e b2 24 fa f7 dc cf ... >
<Buffer 02 36 4e a1 f1 96 2b 3e 0f 2e 26 2e 74 ... >
...
```

Readable Streams

- Inherits from require('stream').Stream
- Events: 'data', 'end', 'close', 'error'
- Methods: pause(), resume(), end(), destroy()
- Property: readable (bool)

Writable Streams

```
var fs = require('fs');
var stream = fs.createWriteStream('/tmp/test.dat');

stream.write(new Buffer('Hello World\n'));
stream.write('How are you?');
stream.end();
```

```
$ node write.js
$ cat /tmp/test.dat
Hello World
How are you?
```

Writable Streams

- Inherits from EventEmitter
- Events: ‘drain’, ‘error’, ‘close’, ‘pause’, ‘resume’
- Methods: write(), end(), destroy()
- Property: writable (boolean)

Buffers

- Default data type for Streams
- Array-like representation of bytes
- Fixed length

Buffers

```
var buffer = new Buffer([1, 10, 255]);
// <Buffer 01 0a ff>
buffer.write('abc');
// <Buffer 61 62 63>
```

Strings

- Ascii, utf-8, binary
- Require copy to send to a socket (slow)

UTF-8

UTF-8

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

http.createServer(function(req, res) {
  var data = '';
  req
    .on('data', function(chunk) {
      data += chunk;
    })
    .on('end', function() {
      console.log('received: ', data);
    });
}).listen(8080);
```

NOT UTF-8 SAFE

UTF-8

- Each character is represented by 1-4 bytes
- Can’t split characters
- Hard to Stream

UTF-8

```
req.setEncoding('utf-8');
```

UTF-8

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

http.createServer(function(req, res) {
   var data = '';
   req.setEncoding('utf-8');
   req
      .on('data', function(buffer) {
        data += buffer;
      })
      .on('end', function() {
        console.log('received: ', data);
      });
}).listen(8080);
```

UTF-8

- 0xxxxxxx (1 byte)
- 110xxxxx (2 bytes)
- 1110xxxx (3 bytes)
- 11110xxx (4 bytes)

UTF-8: String Decoder

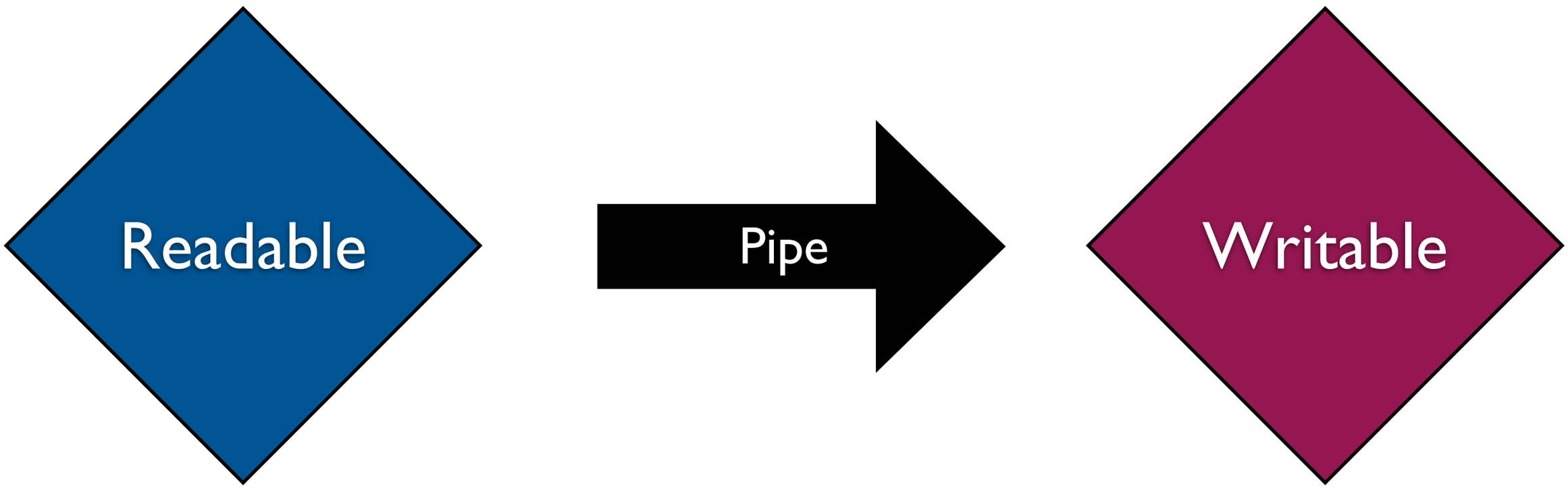
- Scans last 3 bytes of incoming data
- Buffers 1-3 bytes if incomplete character detected
- Only converts/emits the safe part of the string

UTF-8

- UTF-8 is the correct spelling
- utf-8 (lowercase) is ok too (supported by IANA)
- utf8 (no hyphen) is wrong, but supported by node and many other things (browsers)

Pipes

Pipes



File Upload

File

pipe()

- Reads ‘data’ events on source, calls write() on destination

- Also calls end() on destination once source closes

- and more ...

Back Pressure

Back Pressure

- Problem: Writable stream that is slower than the readable stream

- Pipe solves this by calling pause() on the readable stream (if supported)

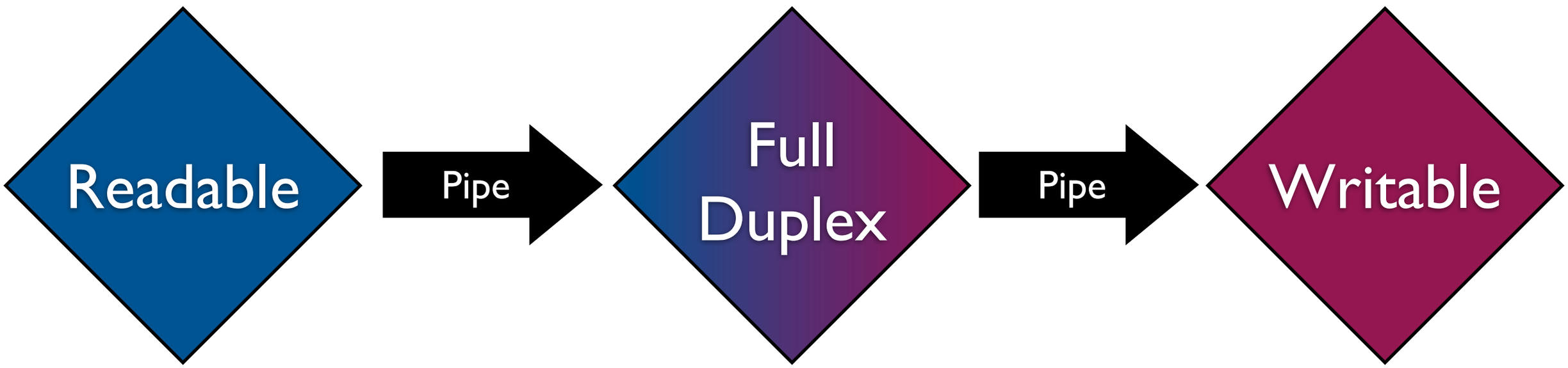
Beware

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

http.createServer(function(req, res) {
  req.pause();
  setTimeout(function() {
    req.on('data', function() { ... });
    req.resume();
  }, 1000);
}).listen(8080);
```

Mega Pipes

Mega Pipes



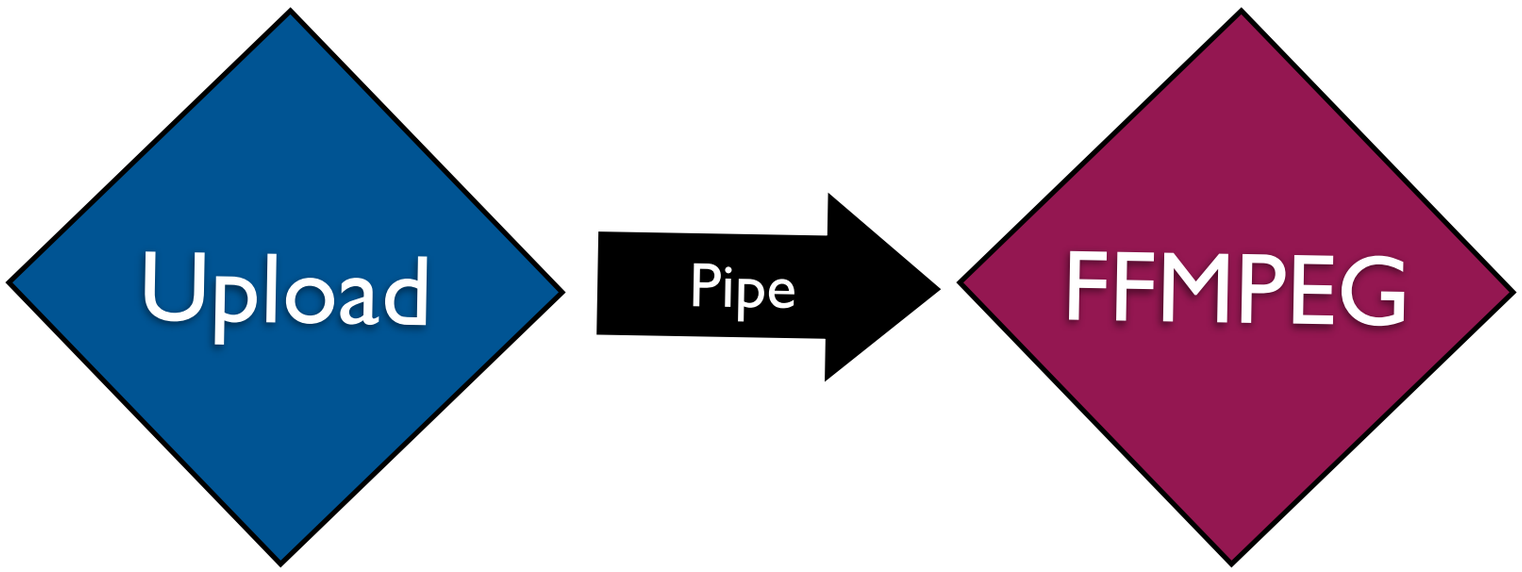
File

Gzip

Socket

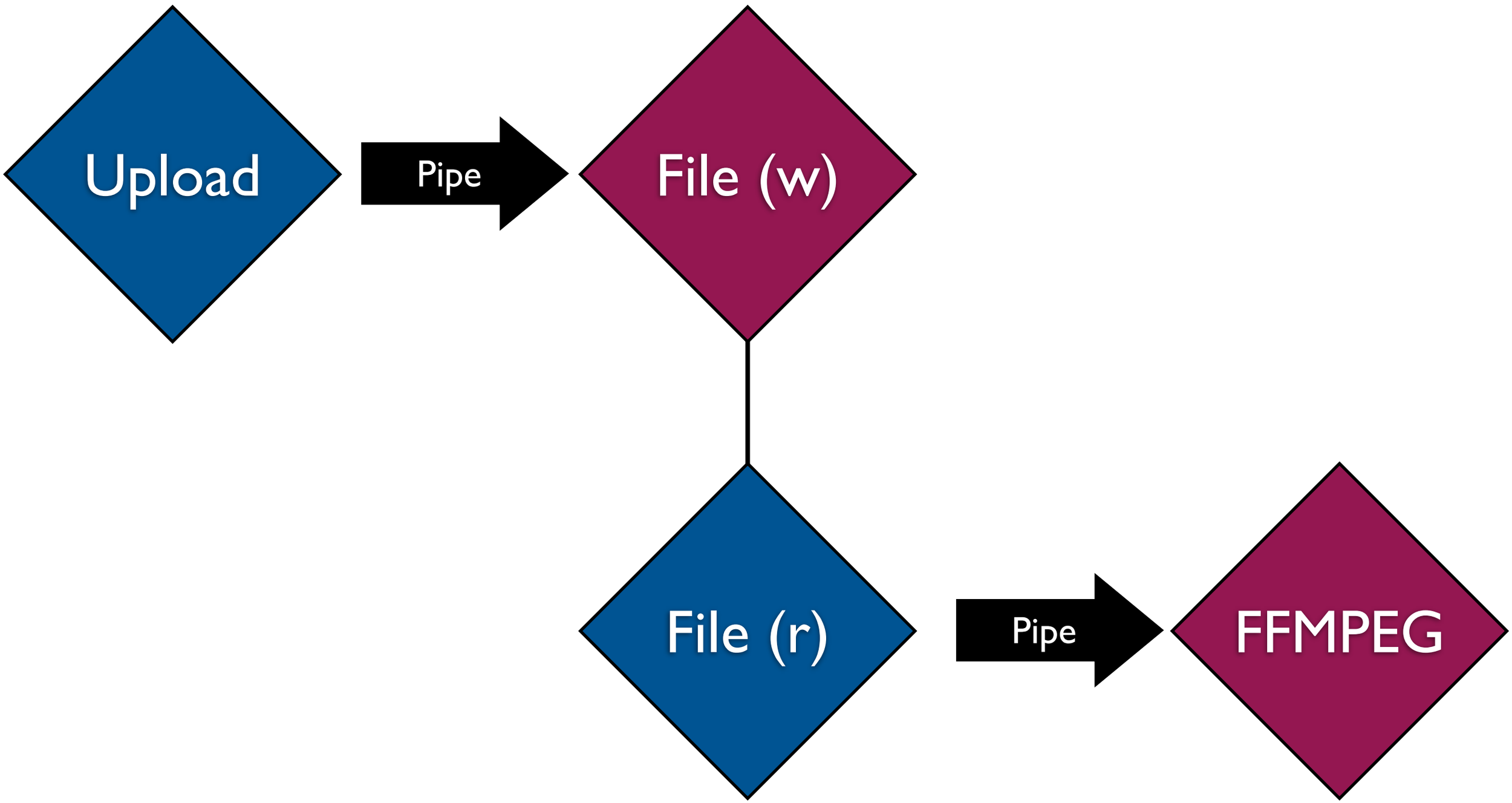
Example: Realtime Encoding

Realtime Encoding



Naive Implementation

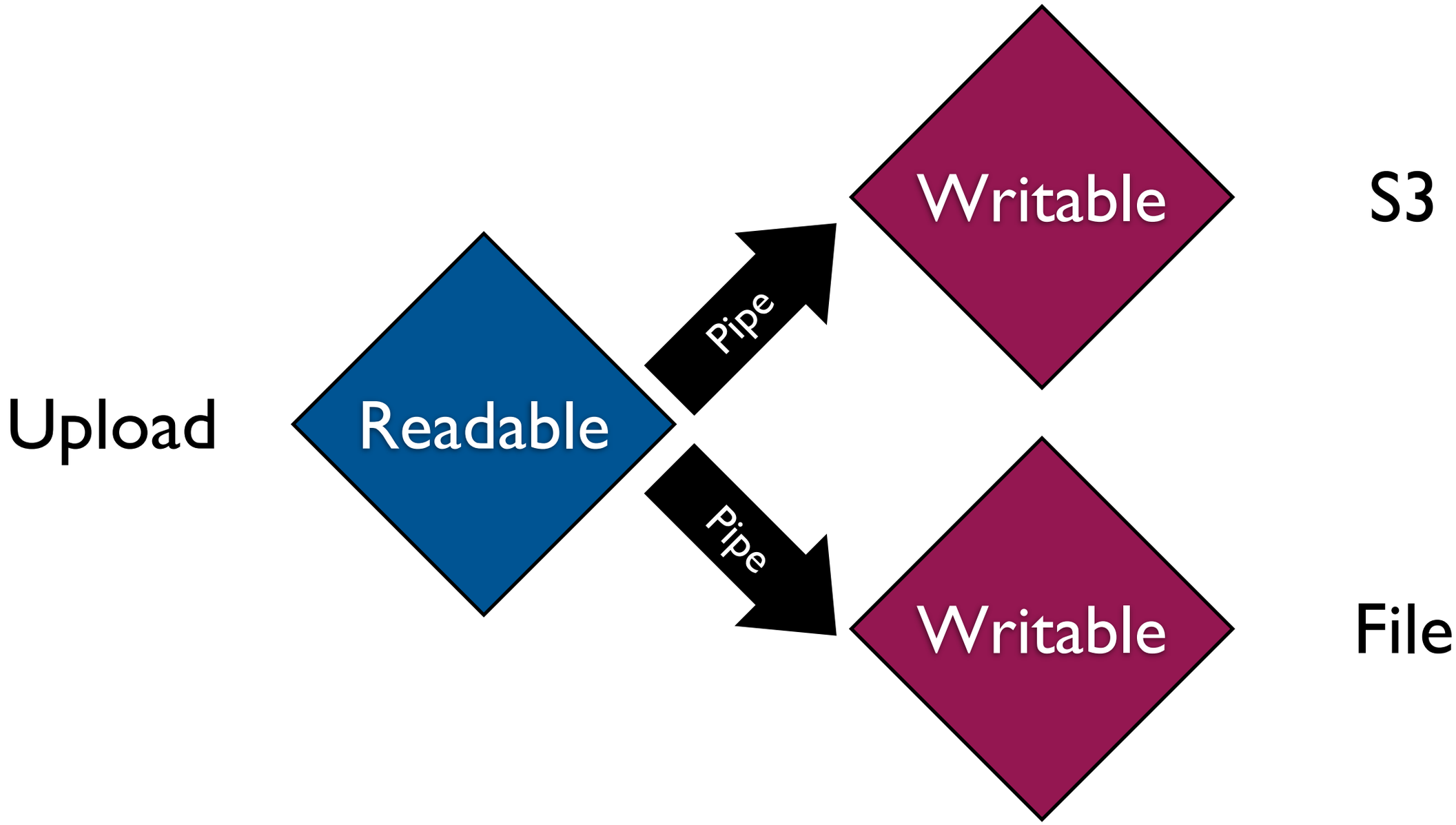
Realtime Encoding



transloadit.com

T-Pipes

T-Pipes



T-Pipes

- One readable stream into 2+ writable streams

- Back-pressure will cause slowest writable stream to determine throughput

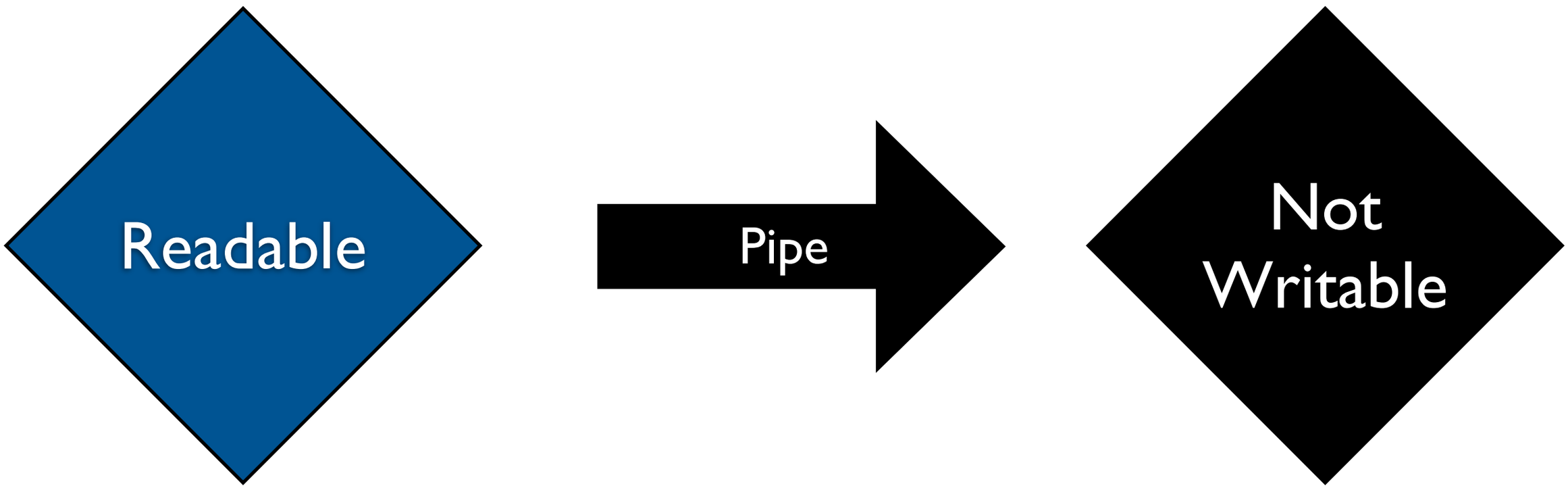
- Also possible to buffer data to avoid back pressure

The Electronic Pipe

NO TAR, NO CARBON MONOXIDE

EPIPE

EPIPE



Writing your own streams

Passthrough Stream

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

module.exports = PassthroughStream;
function PassthroughStream() {
  this.writable = true;
  this.readable = true;
}
util.inherits(PassthroughStream, Stream);

PassthroughStream.prototype.write = function(data) {
  this.emit('data', data);
};

PassthroughStream.prototype.end = function() {
  this.emit('end');
};

PassthroughStream.prototype.destroy = function() {
  this.emit('close');
};
```

Passthrough Stream

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

module.exports = PassthroughStream;
function PassthroughStream() {
  this.writable = true;
  this.readable = true;
}
util.inherits(PassthroughStream, Stream);
```

Passthrough Stream

```
PassthroughStream.prototype.write = function(data) {
  this.emit('data', data);
};

PassthroughStream.prototype.end = function() {
  this.emit('end');
};

PassthroughStream.prototype.destroy = function() {
  this.emit('close');
};
```

Passthrough Stream

```
var PassthroughStream = require('passthrough_stream');
var fs = require('fs');

var source = fs.createReadStream('source.txt');
var dest = fs.createWriteStream('dest.txt');
var passthrough = new PassthroughStream();

source.pipe(passthrough);
passthrough.pipe(dest);

dest.on('close', function() {
  console.log('done!');
});
```

On Github

felixge/node-passthrough-stream

Examples

- delayed stream
- form-data
- growing-file

Questions?

@felixge

NODE.JS CONFERENCE

Brescia - September 24th 2011

www.nodejsconf.it + cfp@nodejsconf.it + @nodejsconfit

< WEB deBS > + GrUSE