Ordered Solutions

Beep Boop

```
console.log('beep boop');
```

Meet Pipe

```
var fs = require('fs');
var file = process.argv[2];
fs.createReadStream(file).pipe(process.stdout);
```

Input Output

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

Transform

```
var through = require('through');
var tr = through(function (buf) {
    this.queue(buf.toString().toUpperCase());
});
process.stdin.pipe(tr).pipe(process.stdout);
```

Lines

```
var through = require('through');
var split = require('split');

var lineCount = 0;
var tr = through(function (buf) {
    var line = buf.toString();
    this.queue(lineCount % 2 === 0
        ? line.toLowerCase() + '\n'
        : line.toUpperCase() + '\n'
    );
    lineCount ++;
});
process.stdin.pipe(split()).pipe(tr).pipe(process.stdout);
```

Concat

```
var concat = require('concat-stream');
process.stdin.pipe(concat(function (src) {
    var s = src.toString().split('').reverse().join('');
    console.log(s);
}));
```

HTTP Server

```
var http = require('http');
var through = require('through');
var server = http.createServer(function (req, res) {
    if (req.method === 'POST') {
        req.pipe(through(function (buf) {
            this.queue(buf.toString().toUpperCase());
        })).pipe(res);
    }
    else res.end('send me a POST\n');
});
server.listen(parseInt(process.argv[2]));
```

HTTP Client

```
var request = require('request');
var r = request.post('http://localhost:8000');
process.stdin.pipe(r).pipe(process.stdout);
```

Websockets

```
var ws = require('websocket-stream');
var stream = ws('ws://localhost:8000');
stream.end('hello\n');
```

HTML Stream

```
var trumpet = require('trumpet');
var through = require('through');
var tr = trumpet();
var loud = tr.select('.loud').createStream();
loud.pipe(through(function (buf) {
    this.queue(buf.toString().toUpperCase());
})).pipe(loud);
process.stdin.pipe(tr).pipe(process.stdout);
```

Duplexer

```
var spawn = require('child_process').spawn;
var duplexer = require('duplexer');
module.exports = function (cmd, args) {
    var ps = spawn(cmd, args);
    return duplexer(ps.stdin, ps.stdout);
};
```

Duplex Redux

```
var duplexer = require('duplexer');
var through = require('through');
module.exports = function (counter) {
    var counts = {};
    var input = through(write, end);
    return duplexer(input, counter);
    function write (row) {
        counts[row.country] = (counts[row.country] || 0) + 1;
    }
    function end () { counter.setCounts(counts) }
};
```

Combiner

```
var combine = require('stream-combiner');
var through = require('through');
var split = require('split');
var zlib = require('zlib');
module.exports = function () {
    var grouper = through(write, end);
    var current;
    function write (line) {
        if (line.length === 0) return;
        var row = JSON.parse(line);
        if (row.type === 'genre') {
            if (current) {
                this.queue(JSON.stringify(current) + '\n');
            }
            current = { name: row.name, books: [] };
        }
        else if (row.type === 'book') {
            current.books.push(row.name);
        }
    }
    function end () {
        if (current) {
            this.queue(JSON.stringify(current) + '\n');
        }
        this.queue(null);
    }
    return combine(split(), grouper, zlib.createGzip());
};
```

Crypt

```
var crypto = require('crypto');
process.stdin
    .pipe(crypto.createDecipher('aes256', process.argv[2]))
    .pipe(process.stdout);
```

Secretz

```
var crypto = require('crypto');
var tar = require('tar');
var zlib = require('zlib');
var through = require('through');
var parser = tar.Parse();
parser.on('entry', function (e) {
    if (e.type !== 'File') return;
    var h = crypto.createHash('md5', { encoding: 'hex' });
    e.pipe(h).pipe(through(null, end)).pipe(process.stdout);
    function end () { this.queue(' ' + e.path + '\n') }
});
var cipher = process.argv[2];
var pw = process.argv[3];
process.stdin
    .pipe(crypto.createDecipher(cipher, pw))
    .pipe(zlib.createGunzip())
    .pipe(parser);
```