Devorando Streams

Primeros pasos con streams en node.js y gulp.js

Streams

Gulp

Approach y Disclaimers

Qué es un stream?

```
ps ax | grep nginx
```

```
tail -f server.log | grep error
```

Qué es un stream?

```
ps ax | grep nginx
```

```
tail -f server.log | grep error
```

Por qué Streams?

Reusabilidad / Composición

Separation of concerns

Asincronismo

Throttling (Backpressure handling)

Dev-Ops

ACK de la naturaleza de I/O

Streams en Node.js

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

var server = http.createServer(function (req, res) {
    fs.readFile(__dirname + '/data.txt',
      function(err, data) {
        res.end(data);
    });
});

server.listen(8000);
```

Streams en Node.js

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

var server = http.createServer(function (req, res) {
    var stream = fs.createReadStream(__dirname + '/
        data.txt');
    stream.pipe(res);
});

server.listen(8000);
```

Streams en Node.js

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

var server = http.createServer(function (req, res) {
    var stream = fs.createReadStream(__dirname + '/
        data.txt');
    stream.pipe(oppressor(req)).pipe(res);
});

server.listen(8000);
```

Que puede ir en un stream?

Buffers

Streams

Objetos

Tipos de Stream

```
fs.createReadStream('data.txt')
            .pipe(oppressor(req))
            .pipe(res)
```

Tipos de Stream

```
fs.createReadStream('data.txt')
            .pipe(oppressor(req))
            .pipe(res)
```

Tipos de Stream

```
fs.createReadStream('data.txt')
            .pipe(oppressor(req))
            .pipe(res)
```

Readable Streams

```
process.stdin.on('readable',
  function () {
    var buf = process.stdin.read(3);
    //more stuff here
```

Readable Streams

```
var Readable = require('stream').Readable;
var rs = Readable();

rs._read = function () {
  var rnd = Math.round(Math.random()*256);
  rs.push(String.fromCharCode(rnd));
}
```

Writable Streams

```
var fs = require('fs');
var ws =
fs.createWriteStream('message.txt');

ws.write('beep ');

setTimeout(function () {
    ws.end('boop\n');
}, 1000);
```

Writable Streams

```
var Writable =
require('stream').Writable;

var ws = Writable();

ws._write = function (chunk, enc, next) {
    console.dir(chunk);
    next();
};
```

Otros Streams

```
var Duplex = require('stream').Duplex;
var ds = Duplex();

ds._write = function (chunk, enc, next) {
// some stuff here

ds._read = funcion() {
// other stuff here
```

Otros Streams

```
var Transform =
require('stream').Transform;

var ts = Transform();

ts._transform = function (chunk, enc,
next) {
// some stuff here that can do push

ts._flush = function (done) {
// stuff for when the write part is done
```

Pasado y Futuro

"It's a jungle out there"

"v3 is coming"

Through(2)

```
var truncate = through2(
  //[options],
  function (chunk, encoding, callback) {
    this.push(chunk.slice(0, 10))
    return callback()
  }
  //, [flush function]
)
```

Functional Streams

```
var truncate = through2-map(
  function(chunk) {
    return chunk.slice(0, 10)
  }
)
```

Gulp.js

"The streaming build system"

Gulp.js

100% asíncrónico por default

Gulp.js

Code over configuration

Gulp.js

```
gulp.task('jade', function() {

  gulp.src('./client/templates/*.jade')
  .pipe(jade())
  .pipe(minify())
  .pipe(gulp.dest('./build/templates'));

});
```

Gulp.js

```
gulp.task('jade', function() {

  gulp.src('./client/templates/*.jade')
  .pipe(jade())
  .pipe(minify())
  .pipe(gulp.dest('./build/templates'));

});
```

Gulp.js

```
gulp.task('jade', function() {

  gulp.src('./client/templates/*.jade')
  .pipe(jade())
  .pipe(minify())
  .pipe(gulp.dest('./build/templates'));

});
```

Gulp.js

```
gulp.task('jade', function() {

  gulp.src('./client/templates/*.jade')
  .pipe(jade())
  .pipe(minify())
  .pipe(gulp.dest('./build/templates'));

});
```

Gulp.js

```
gulp.task('default', function() {

   gulp.watch('./client/templates/*.jade'
      ,['jade']);

});
```

Vinyl

Virtual file system que provee:

```
.options (que incluye contents)
.isBuffer()
.isStream()
.isNull()
.pipe()
```

DEMO TIME!

Conclusiones

No escriban plugines de gulp sin antes preguntarse:

Hay un plugin?

Hay un package de node?

Es una task (Userland)?

Conclusiones

Trabajar con Streams es trabajar con Archivos

Trabajar con Streams es trabajar con “Arrays”

Conclusiones

Acostumbrate al caos y a leer código

Conclusiones

Reusar lo que ya anda:

pipe antes que read-write

libs que wrappean el core

Conclusiones

El asincronismo no es gratis

Conclusiones

Gulp tiene una postura muy opinionada - Take it or leave it

Referencias

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

"How streams help to raise node.js performance"
https://www.youtube.com/watch?v=QgEuZ52OZtU

"Why you shouldn't create a gulp plugin"
http://blog.overzealous.com/post/74121048393/why-you-shouldnt-create-a-gulp-plugin-or-how-to-stop