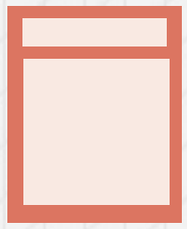
STREAMS

- LEVEL THREE -

WHAT ARE STREAMS?



Start Processing Immediately

Streams can be readable, writeable, or both

The API described here is for streams in Node version v0.10.x a.k.a. streams2

STREAMING RESPONSE

readable stream *writable stream*

```
http.createServer(function(request, response) {
  response.writeHead(200);
  response.write("<p>Dog is running.</p>");
  setTimeout(function(){
    response.write("<p>Dog is done.</p>");
    response.end();
  }, 5000);
}).listen(8080);
```

Our browser receives

"Dog is running."

(5 seconds later)

"Dog is done."

STREAMS

HOW TO READ FROM THE REQUEST?

Let's print what we receive from the request.

```
http.createServer(function(request, response) {
  response.writeHead(200);
  request.on('readable', function() {
    var chunk = null;
    while (null !== (chunk = request.read())) {
      console.log(chunk.toString());
    }
  });
  request.on('end', function() {
    response.end();
  });
}).listen(8080)
```

HOW TO READ FROM THE REQUEST?

```
http.createServer(function(request, response) {
   response.writeHead(200);
   request.on('readable', function() {
      var chunk = null;
      while (null !== (chunk = request.read())) {
         response.write(chunk);
      }
   });
   request.on('end', function() {
      response.end();
   });
}).listen(8080)
```

```
request.pipe(response);
```

LET'S CREATE AN ECHO SERVER!

```
http.createServer(function(request, response) {
  response.writeHead(200);
  request.pipe(response);
}).listen(8080)
```

```
$ curl -d 'hello' http://localhost:8080
```

----> Hello *on client*

Kinda like on the command line

```
cat 'bleh.txt' | grep 'something'
```

STREAMS

DOCUMENTATION http://nodejs.org/api/

Stability Scores

File System

> Stability: 3 - Stable

File I/O is provided by simple wrappers around standard PO
`require('fs')`. All the methods have asynchronous and sy

The asynchronous form always take a completion callback as
completion callback depend on the method, but the first argu
operation was completed successfully, then the first argumen

When using the synchronous form any exceptions are immed
exceptions or allow them to bubble up.

Here is an example of the asynchronous version:

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

fs.unlink('/tmp/hello', function (err) {
```

Stream

> Stability: 2 - Unstable

A stream is an abstract interface implemented by various objects in No
server is a stream, as is stdout. Streams are readable, writable, or both.

You can load the Stream base classes by doing `require('stream')`. T
Readable streams, Writable streams, Duplex streams, and Transform s

This document is split up into 3 sections. The first explains the parts of
use streams in your programs. If you never implement a streaming API

The second section explains the parts of the API that you need to use i
yourself. The API is designed to make this easy for you to do.

The third section goes into more depth about how streams work, inclu
functions that you should probably not modify unless you definitely kn

API for Stream Consumers

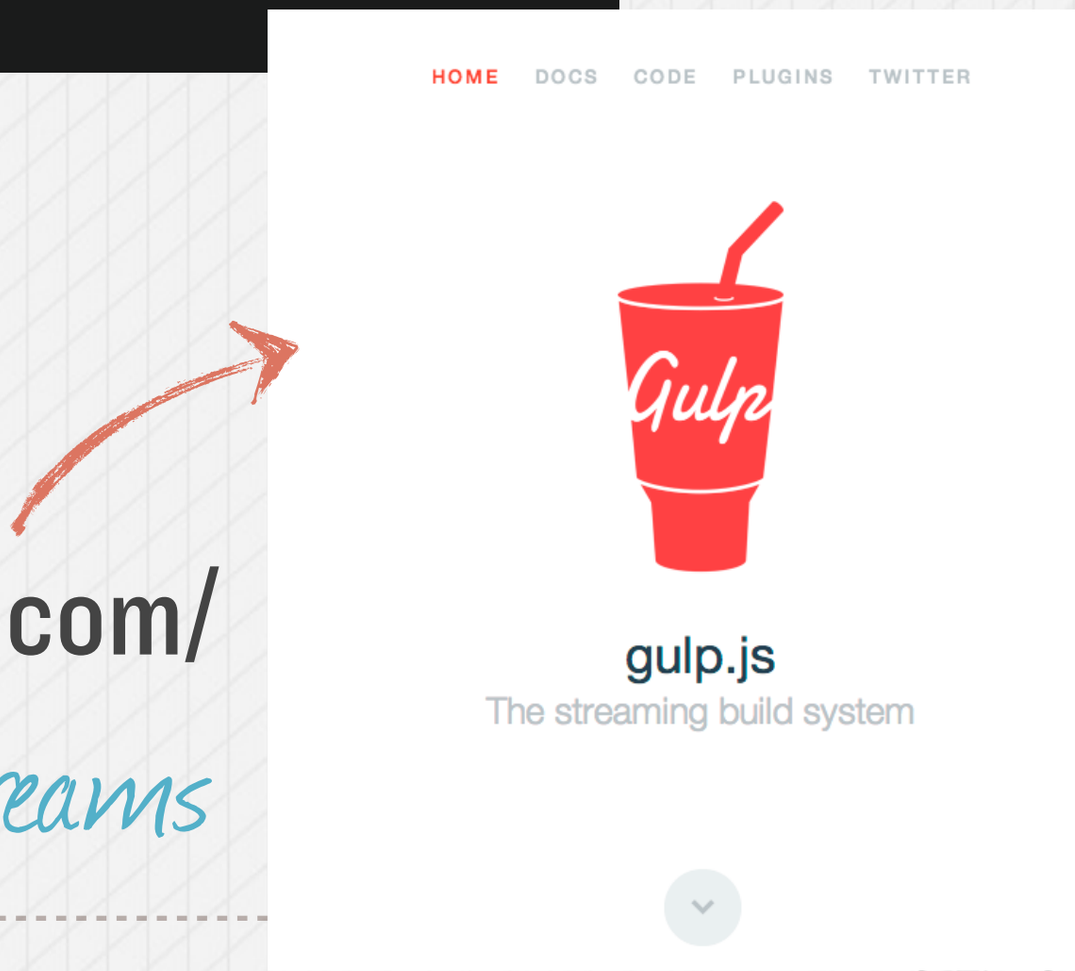
STREAMS

READING AND WRITING A FILE

```
var fs = require('fs');
var file = fs.createReadStream("readme.md");
var newFile = fs.createWriteStream("readme_copy.md");

file.pipe(newFile);
```

require filesystem module



http://gulpjs.com/

Build system built on top of Streams

STREAMS

UPLOAD A FILE

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

http.createServer(function(request, response) {
   var newFile = fs.createWriteStream("readme_copy.md");
   request.pipe(newFile);

   request.on('end', function() {
      response.end('uploaded!');
   });
}).listen(8080);
```

```
$ curl --upload-file readme.md http://localhost:8080
```

----> uploaded!

STREAMS

THE AWESOME STREAMING

server

client

storage

original file

transferred file

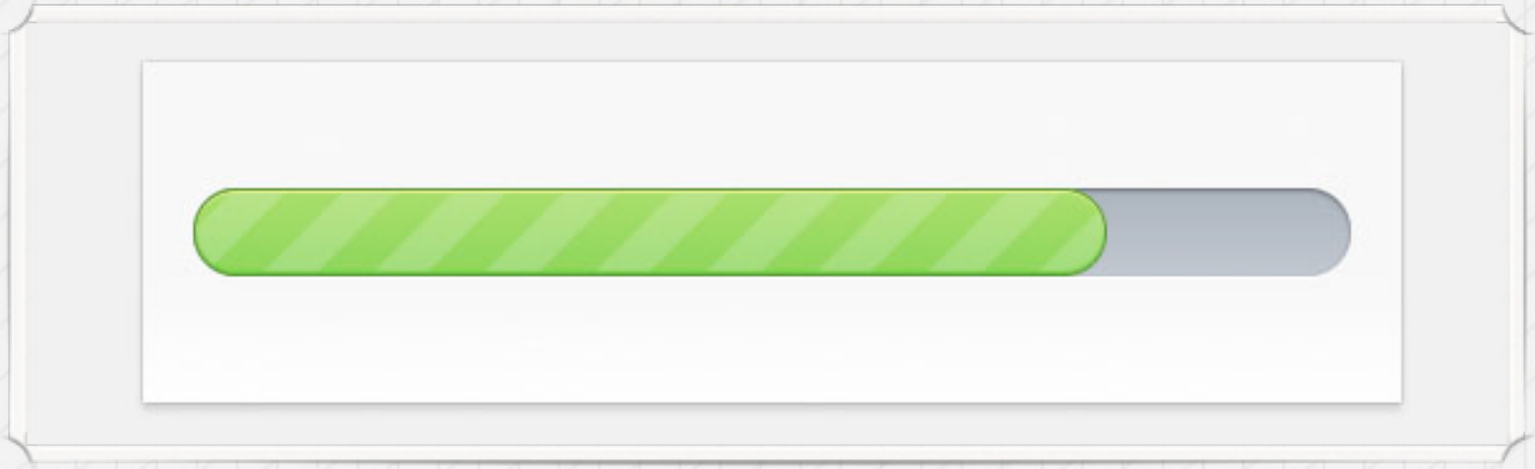
non-blocking

0s

5s

10s

FILE UPLOADING PROGRESS



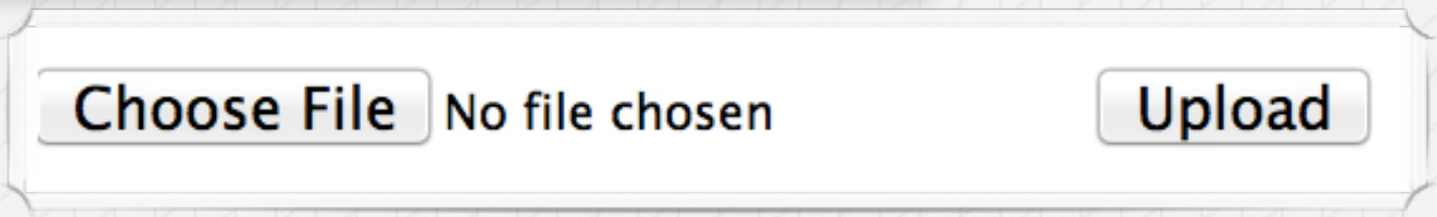
STREAMS

FILE UPLOADING PROGRESS

```
$ curl --upload-file file.jpg http://localhost:8080
```

Outputs:

```
progress: 3%
progress: 6%
progress: 9%
progress: 12%
progress: 13%
...
progress: 99%
progress: 100%
```



We're going to need:

- HTTP Server
- File System

STREAMS

REMEMBER THIS CODE?

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

http.createServer(function(request, response) {
   var newFile = fs.createWriteStream("readme_copy.md");
   request.pipe(newFile);

   request.on('end', function() {
      response.end('uploaded!');
   });
}).listen(8080);
```

STREAMS

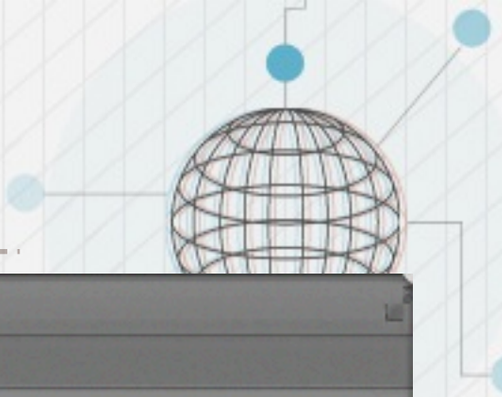
REMEMBER THIS CODE?

```
http.createServer(function(request, response) {
   var newFile = fs.createWriteStream("readme_copy.md");
   var fileBytes = request.headers['content-length'];
   var uploadedBytes = 0;

   request.on('readable', function() {
     var chunk = null;
     while(null !== (chunk = request.read())){
       uploadedBytes += chunk.length;
       var progress = (uploadedBytes / fileBytes) * 100;
       response.write("progress: " + parseInt(progress, 10) + "%\n");
     }
   });

   request.pipe(newFile);
   ...
}).listen(8080);
```

SHOWING PROGRESS



STREAMS