STREAM IN NODE.JS

Node Sydney October 2018

WHO AM I ?

- Ashwin Kandel (@incessantmeraki)
- Works at Quantium
- Works with Node.js, React, & AWS

WHAT IS NODE.JS ?

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

WHAT ACTUALLY IS NODE.JS ?

I/O interface

WHAT DOES THAT MEAN ?

Helps write program in JavaScript that talks to

COMON THING FOR ALL I/O ?

This is where 👉 Stream 👈 comes into the picture

WHAT IS STREAM ?

Stream is a handy abstraction that's there in node.js to manipulate data

IS STREAM A NEW CONCEPT ?

Been there since the days of UNIX in form of pipe

```
$ cat file | wc -c
```

WHY STREAMS ?

EXAMPLE: HANDLING LARGE AMOUNT OF DATA

- Serve a large file over HTTP server
- w/o Stream
- with Stream
- See the difference

TYPES OF STREAM

- Readable : produces data
- Writable : consumes data
- Transform : consumes data , modifies it, and produces the modified thing
- Duplex : produces and consumes data separately. Similar to telephone

PIPE

used to connect different streams

- Readable: `readable.pipe(X)`
- Writeable: `X.pipe(writeable)`
- Transform: `X.pipe(transform).pipe(Y)`
- Duplex: `X.pipe(duplex).pipe(X)`

READABLE STREAM METHODS

- `stream.on('readable', callback)`
- `stream.on('end' ,callback)`
- `stream.on('data', callback)`
- `stream.read()`
- `stream.pipe(writeableStream)`

FLOW MODE V/S PULL MODE

WRITEABLE STREAM METHODS

- `stream.write(data)`
- `stream.on('finish', callback)`
- `stream.end(), stream.end(data)`
- `x.pipe(stream)`

EXAMPLE: EXPLORING COMMON METHODS AVAILABLE

- Create Readable and Writeable Stream to files
- Use methods shown below to read and write data

BUILT-IN STREAMS

- process
- http
- zlib
- crypto

EXAMPLES:

process

```
process.stdin //readable stream
process.stdout //writeable stream
```

http

```
http.createServer(function(req,res) {...})
// req = readable stream and res = writeable stream

http.request(opts, function(res){...})
//res = readable stream
```

zlib

```
var zlib = require('zlib')

zlib.createGzip(opts) // transform stream to compress with gzip
zlib.createGunzip(opts) // transform stream to uncompress with gzip
zlib.createDeflate(opts) // transform stream to compress with deflate
zlib.createDeflateRaw(opts) // transform stream to compress with raw deflate
zlib.createInflate(opts) // transform stream to uncompress with deflate
zlib.createInflateRaw(opts) // transform stream to uncompress with raw deflate
zlib.createUnzip(opts) // transform stream to uncompress gzip and deflate
```

crypto

```
var crypto = require('crypto')

crypto.createCipher(algo, password) // transform stream to encrypt
crypto.createDecipher(algo, password) // transform stream to decrypt
crypto.createCipheriv(algo, key, iv) // transform stream to encrypt with iv
crypto.createDecipheriv(algo, key, iv) // transform stream to decrypt with
 iv
crypto.createHash(algo) // transform stream to output cryptographic hash
crypto.createHMAC(algo, key) // transform stream to output HMAC digest
crypto.createSign(algo) // writable stream to sign messages
crypto.createVerify(algo) // writable stream to verify signatures
```

TCP

```
//Server
var net = require('net')
var server = net.createServer(function(stream){})
server.listen(5000)

//Client
net.connect(5000, localhost)
```

Split

```
var split = require('split2')
var fs = require('fs')

fs.createReadStream('file')
  .pipe(split())


fs.createReadStream('file')
  .pipe(split())
  .pipe(process.stdout)
```

CREATING STREAMS

Use user-land modules to create streams

from2

```
var from = require('from2')

var rStream = from(function(size, next) {})
```

to2

```
var to =  require('to2')

var wStream = to2(function(buffer, next) {})
```

through2

```
var through = require('through2')

var tStream = through(function (buff, enc, next) {...})
```

duplexify

```
var duplexify = require('duplexify')

// turn writableStream and readableStream into a single duplex stream
var dup = duplexify(writableStream, readableStream)

dup.write('hello world') // will write to writableStream
dup.on('data', function(data) {
  // will read from readableStream
})
```

MISCELLANEOUS MODULES

collect-stream

end-of-stream

pumpify

websocket-stream

Stream abstraction works on the browser too, use browserify

FUTHER INFORMATION....

- Try nodeschool workshop **stream-adventure**
- Read **stream-handbook** by @substack
- Follow works of awesome people like @mafintosh, @substack, @yoshuawuyts...

THE END

Question time...