Working with Streams

Samer Buna

@samerbuna www.jscomplete.com

“Streams are Node's best and most misunderstood idea”

Dominic Tarr

Streams

Readable Streams	Writable Streams
HTTP responses, on the client	HTTP requests, on the client
HTTP requests, on the server	HTTP responses, on the server
fs read streams	fs write streams
zlib streams	zlib streams
crypto streams	crypto streams
TCP sockets	TCP sockets
child process stdout and stderr	child process stdin
process.stdin	process.stdout, process.stderr

What are streams?

Streams

Collections of data that might not be available all at once and don't have to fit in memory.

Types of Streams

Readable

fs.createReadStream

Writable

fs.createWriteStream

Duplex

net.Socket

Transform

zlib.createGzip

All streams are

EventEmitters

```
src.pipe(dst);
```

Piping

Linux

```
a | b | c | d
```

Node.js

```
a.pipe(b).pipe(c).pipe(d);
```

```
a.pipe(b);
b.pipe(c);
c.pipe(d);
```

Streams

Implementing

require('stream')

Consuming

piping/events

Readable Streams

Events

- data
- end
- error
- close
- readable

Functions

- pipe(), unpipe()
- read(), unshift(), resume()
- pause(), isPaused()
- setEncoding()

Writable Streams

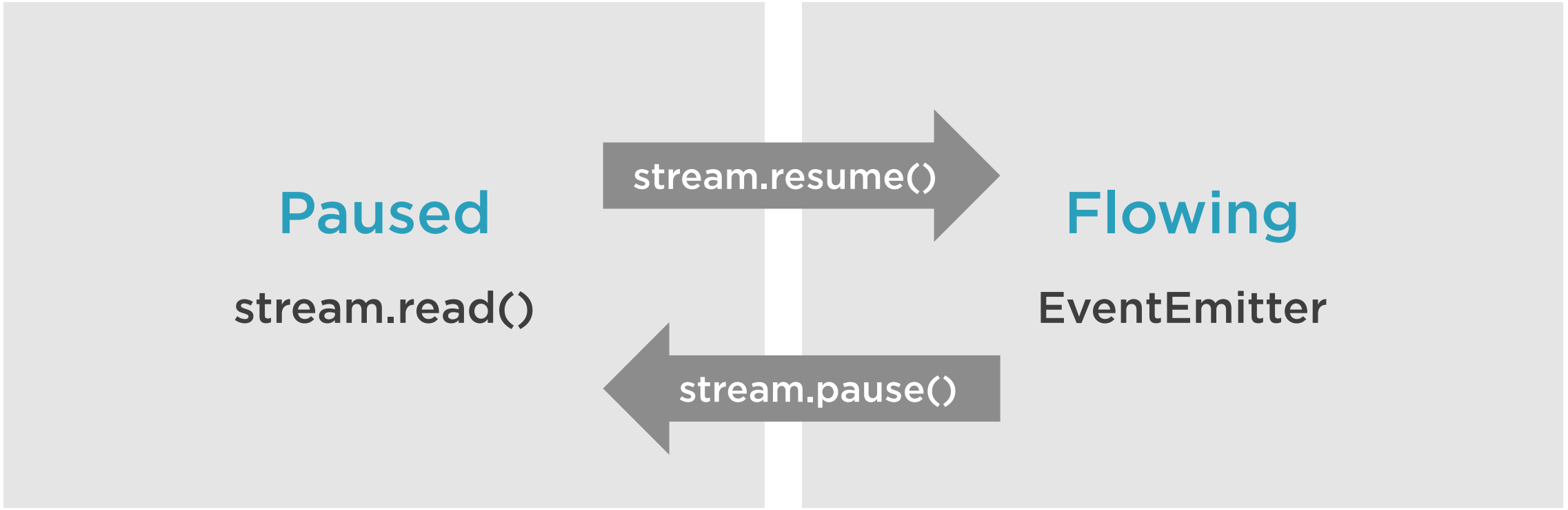
Events

- drain
- finish
- error
- close
- pipe/unpipe

Functions

- write()
- end()
- cork(), uncork()
- setDefaultEncoding()

Readable Streams



Summary

Composability with streams

Stream types

- pipe() vs events
- Implementing vs consuming
- Paused vs flowing

new

- sream.Readable
- stream.Writable
- stream.Duplex
- stream.Transform

The zlib/crypto transform streams