WEB DAY 2023

MILANO 16 MARZO

The last 5 years of streams in Node.js

Paolo Insogna

Node.js Core, OramaSearch, NearForm

/* Sponsor */

webscience an adesso company

.NET on aws

improove

/* Partner */

Cloud Champions

Serverless Italy

TOMORROWDEVS

TORINO

AWS User Groups Milano

{codemotion}

BLAZOR DEVELOPER ITALIANI

tag Talent Garden

ACCESSIBILITY DAYS

AZURE MEETUP MILANO

POINTER PODCAST

View online

Download PDF

The last 5 years of streams in Node.js

Paolo Insogna Node.js Core, OramaSearch, NearForm

NearForm

Panta rei!

Hello, I'm Paolo!

Node.js	Core Member
OramaSearch	Co-founder & Principal Architect
NearForm	Staff Developer Experience Engineer

https://cowtech.it **p_insogna** **ShogunPanda**

WE'RE BOLD
WE'RE FLEXIBLE
WE'RE OPEN
WE'RE EMPOWERING

Global Delivery Org with 300+ and counting

Major Contributors to the Open Source Web Platform

NPM monthly downloads

1B

Represents modules used globally

8%

We are hiring!

nearform.com | **follow us on**

NearForm

orama

search, everywhere

A **resilient**, **innovative** and **open-source** search experience to achieve seamless integration with your infrastructure and data

@OramaSearch

What are streams anyway?

Pretty simple, isn't it?

“A stream is an abstract interface for working with streaming data in Node.js.”

Node.js official documentation

They are powerful

Fully asynchronous

Since they are based on events and async I/O, the event loop is not blocked.

Minimum memory footprint

Only single chunks of data are processed instead of loading the full data in memory.

This will return!

Expandable

There are tons of away to implement new stream.

Can I use it in the browser?

Meet readable-stream

It is a NPM package which mirrors the Node.js stream module and additionally adapts it to use in the browsers and/or bundlers.

https://www.npmjs.com/package/readable-stream

How is the package built?

1. **Download the code**
   Node.js source code is downloaded.

2. **Copy the stream module**
   The entire stream module is copied into the destination directory.

3. **Manipulate the source**
   All references to other node modules are removed.

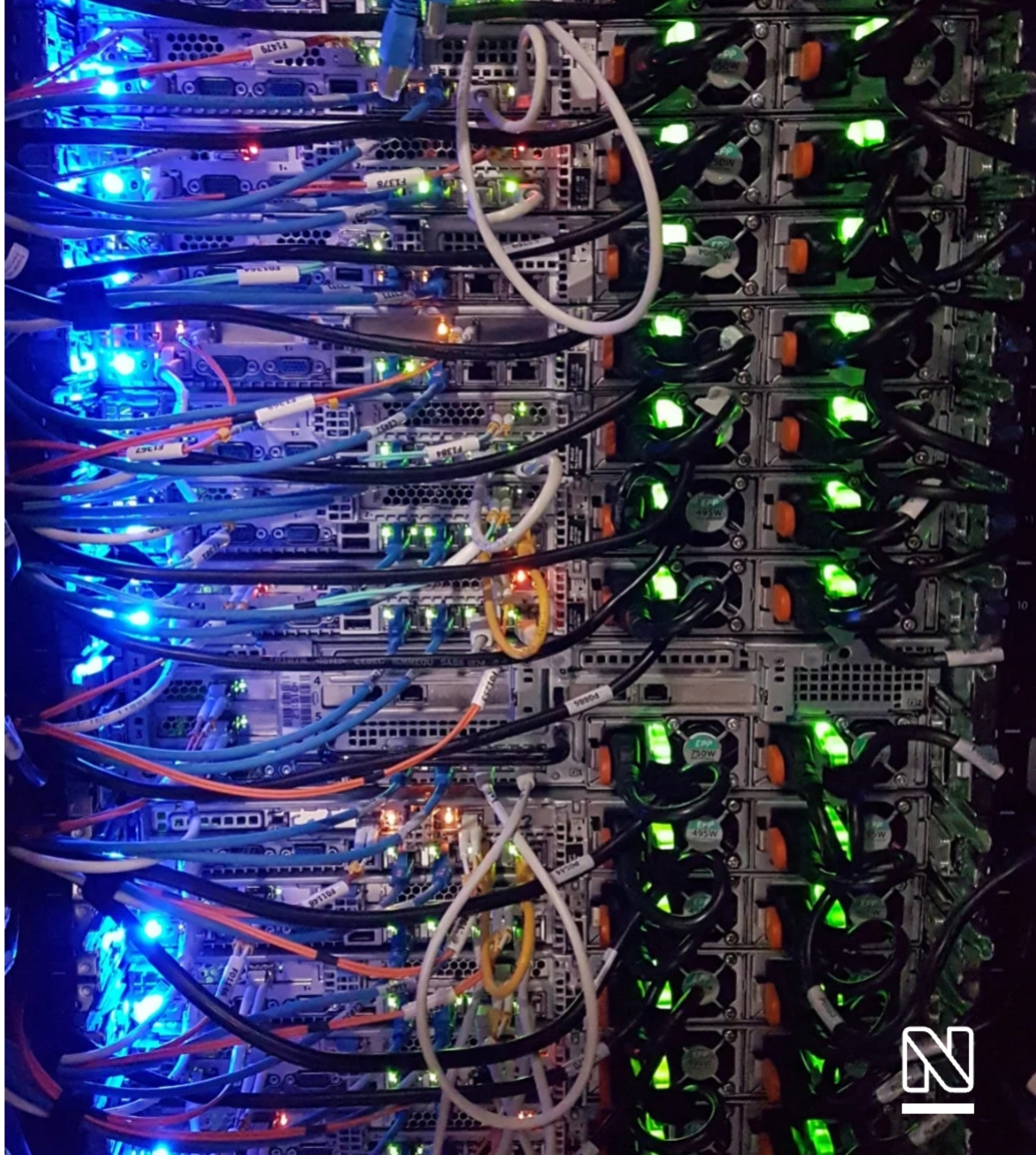
4. **Add custom modules**
   Custom and platform-agnostic version of the needed modules re-inserted in the distribution.

Project status

Latest series is 4.x.x.
It was initally released on **June 14th, 2022**, three years after the last one.

Update of the stream module
It mirrors the module in Node.js **18.x.x** which is the current LTS version.

Long overdue
Series 3.x.x shipped the module present in Node.js **10.x.x** (2018).



Sorry for the long wait!

What has changed in streams since then?

Broader status handling

An entire new set of properties for readable stream
`readableAborted`, `readableDidRead`, `readableEncoding` and `readableObjectMode`

Tracking the end **event**
`stream.Readable.readableEnded` and `stream.Writable.writableEnded`

Tracking flushing status
`stream.Writable.writableFinished`

More predictable event flow

Be reasonable

The `autoDestroy` option is now enabled by default: after a `end` or `finish` event the stream will call `destroy` automatically (and only once).

Less events chain

The `end` or `finish` events are not emitted if a `error` event was emitted.
Emitting a `close` event before the `end` event is now considered an error.

Do not be eager

`stream.pipeline` will wait for unfinished streams.

... so boring ...

Let's see the exciting parts!

Stream from iterables

`stream.Readable.from` is a new function that allow to create a stream from any object implementing the (async) iterator protocol.

```
import Readable from 'readable-stream'

async function* generate() {
  yield 'hello'
  yield 'streams'
}

const readable = Readable.from(generate())
```

Duplex stream from anything

`stream.Duplex.from` can now be used to create a duplex stream from something else.

Let's play a little game!

```
import Duplex from 'readable-stream'

Duplex.from([
  new Promise((resolve) => setTimeout(resolve('1'), 1500)),
  new Promise((resolve) => setTimeout(resolve('2'), 1500)),
  new Promise((resolve) => setTimeout(resolve('3'), 3000))
]);
```

Promises API

A new module exposes was added to Node.js: `stream/promises`.

It exposes a promises based API version of `stream.finished` and `stream.pipeline`.

```js
import { createReadStream, createWriteStream } from 'node:fs'
import { createGzip } from 'node:zlib'
import { Stream } from 'readable-stream'

const { finished, pipeline } = Stream.promises

await pipeline(
  createReadStream('archive.tar'),
  createGzip(),
  createWriteStream('archive.tar.gz')
)

const rs = createReadStream('archive.tar')
await finished(rs.resume())
```

Functional style helpers

You can manipulate readable streams using common functional style helpers.

Manipulating
`filter`, `map`, `reduce`, `forEach`, `flatMap`

Searching
`some`, `find`, `every`

Fetching
`toArray`

With great power comes great responsibility™

The `toArray` method **reads the entire stream into memory**, which is exactly what streams are meant to avoid.

Promise me you will only use it when absolutely necessary!

What is used under the hood?

Build toolchain

Compared to the previous one it still uses regular expression but it is much smaller thanks to a more modern Javascript environment.

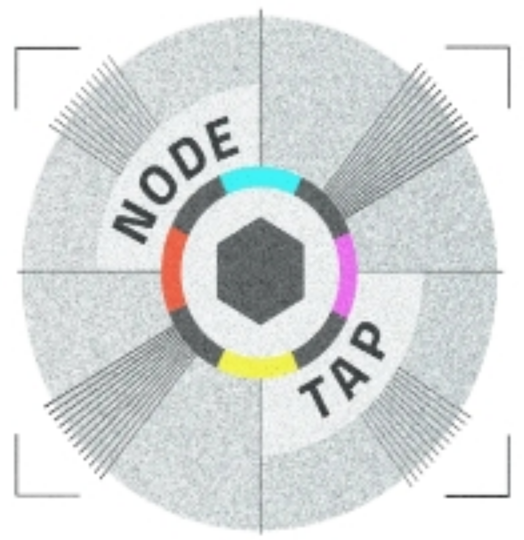
ESM

Babel

Prettier

Testing technologies

A complex testing environment to match most desired scenarios across Node.js version, browsers and bundlers.



TAP
Used to test in the Node.js environment.

Tape
Along with custom runner and parser, it is used to test browsers.

Playwright
It used to easily test in all major browsers.

We test 100 configuration in the CI!

... that's all folks!™

There are obviously more changes, but these were the most juicy.

Remember to thank these guys!

Obviously I haven't done all this by myself.

Matteo Collina
https://github.com/mcollina

Robert Nagy
https://github.com/ronag

Benjamin Gruenbaum
https://github.com/benjamingr

One last thing™

“The man who is swimming against the stream knows the strength of it.”

Woodrow Wilson

Questions?

Thank you!

Paolo Insogna
Node.js Core, OramaSearch, NearForm

@p_insogna
paolo.insogna@nearform.com

NearForm

WEB DAY 2023

MILANO 16 MARZO

GRAZIE