



View online



Download PDF

# Node.js HTTP parser, what's going on?

**Paolo Insogna**

Node.js TSC, Principal Engineer @ **Platformatic**

**Nothing  
beats a classic!**



# Hello, I'm **Paolo!**



**Node.js**

Technical Steering Committee Member

**Platformatic**

Principal Engineer



[paoloinsogna.dev](https://paoloinsogna.dev)



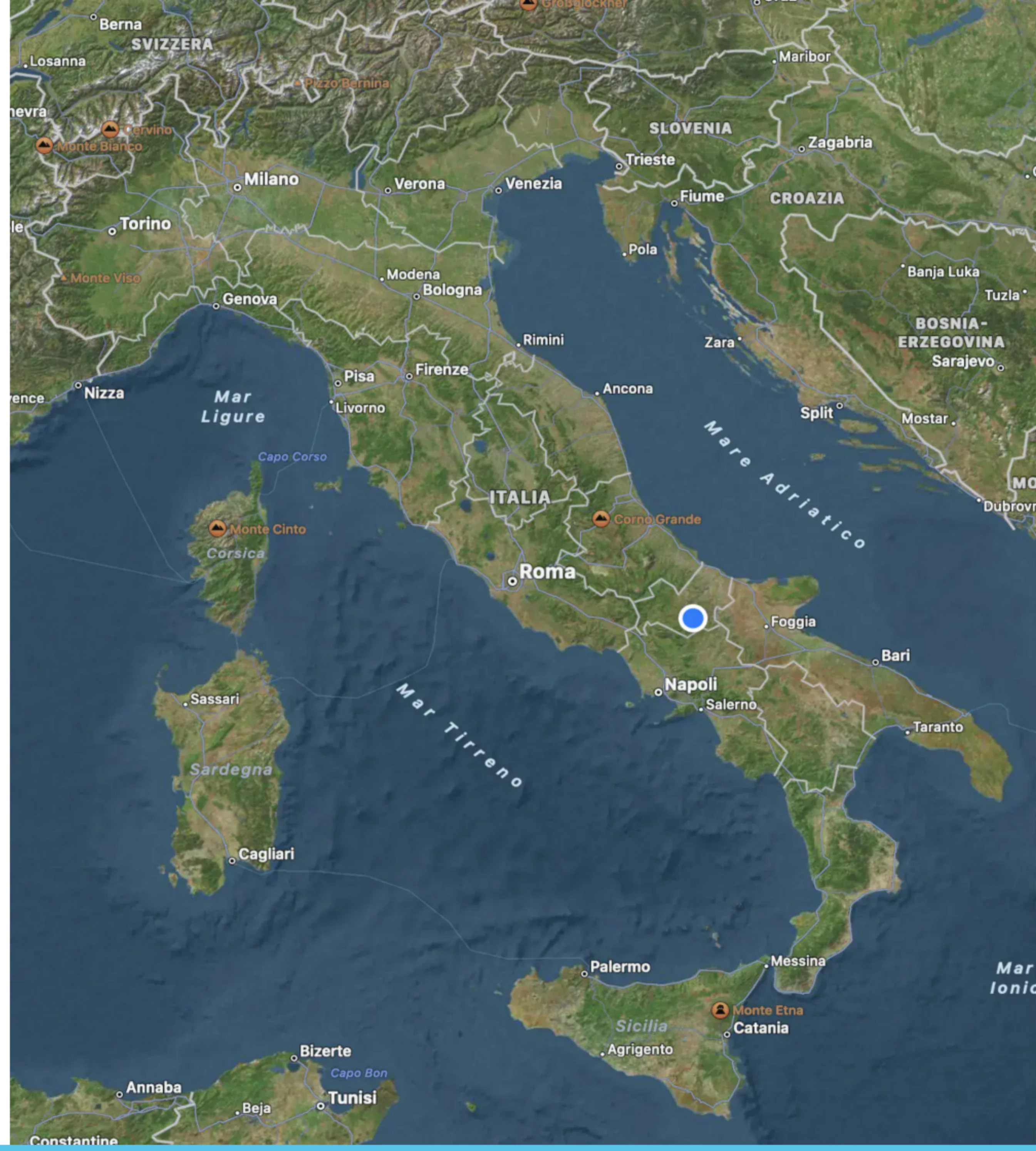
[ShogunPanda](#)



[p\\_insogna](#)



[pinsogna](#)



**We all love HTTP!**



**Which HTTP  
are you?**



# The choice is narrow

Even if the HTTP protocol is more than 30 years old, only **three** current versions of it exist as of today. The others are considered obsolete.

**1** **HTTP/1.1**  
The last version of the initial protocol. By far the most famous and most used.

**2** **HTTP/2**  
Developed on top of SPDY to remove some problems of HTTP.

**3** **HTTP/3**  
Developed on top of QUIC to solve TCP problems.

# HTTP/1.1: Can't beat a classic



## **The grandparent**

Developed in 1989 and by far the mostly used (80% of websites).



## **Textual protocol**

Parsing can be easily implemented in any language.



## **Keep-Alive and FIFO pipelining**

A single TCP connection can be reused.

# HTTP/2: Don't break too much



## **The parent**

Development started in 2012 and end up being standardized in 2015.



## **Baby-steps**

The semantics are the same, only the TCP connection usage has changed.



## **Multiplexing and real parallelism**

It uses a SPDY derived communication protocol for better multiplexing.



# HTTP/3



## **The sibling**

Development started in 2018 and end up being standardized in 2022.



## **Farewell, TCP!**

QUIC is now used, which means HTTP has switched to UDP.



## **No head of line blocking**

Using multiplexing over UDP minimizes problems introduced by lost packets.

# What about Node.js?

**1+2**

**HTTP/1.1 and HTTP/2**

Node.js has a stable implementation.

**3**

**HTTP/3**

Work in progress, **stay tuned!**

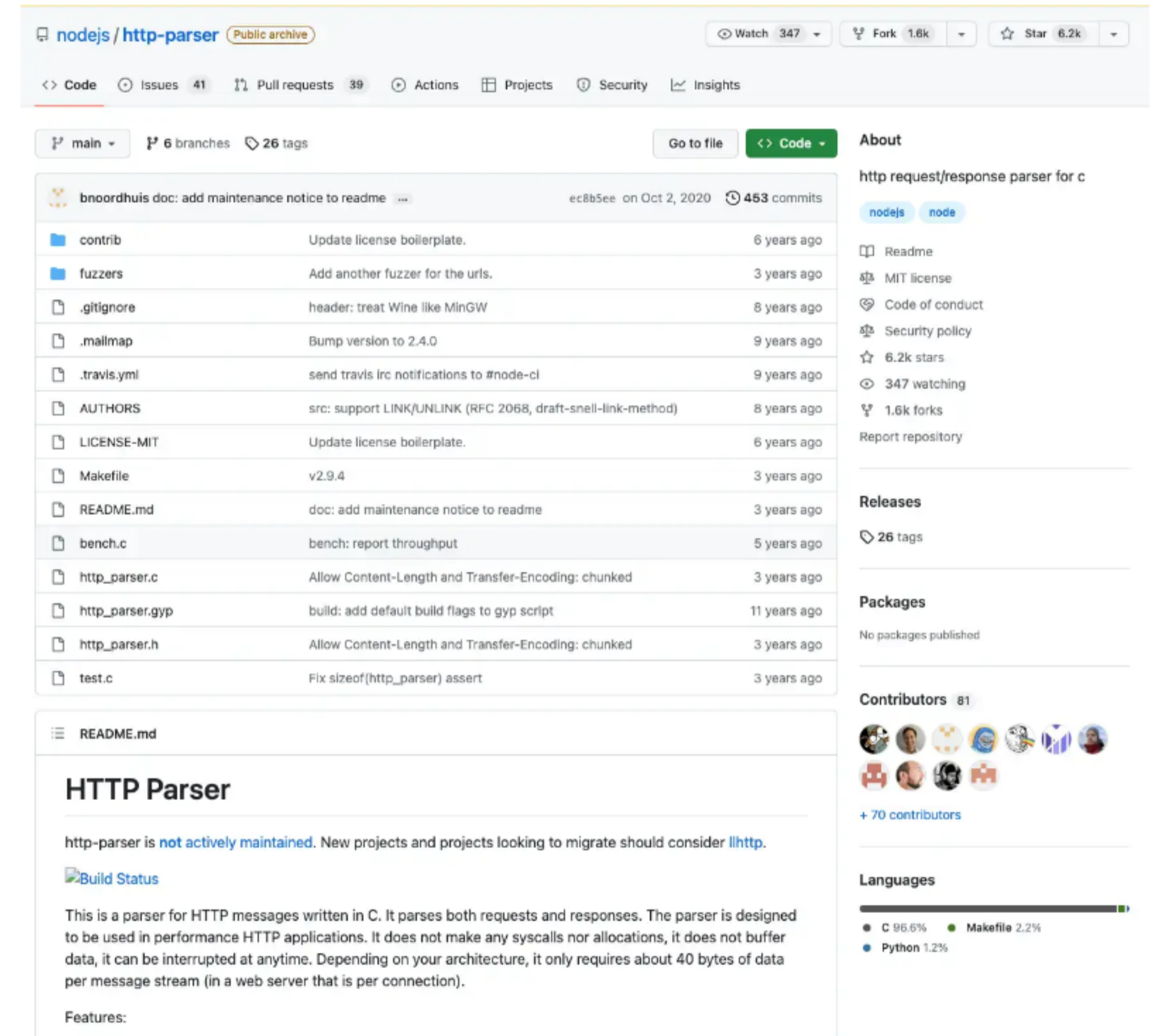
**Let's focus!**



# The original parser

`http_parser` was the original HTTP/1.1 parser of Node.js

It existed since the early days of Node.js and it was one of its older dependencies.



nodejs / http-parser Public archive

Watch 347 Fork 1.6k Star 6.2k

Code Issues 41 Pull requests 39 Actions Projects Security Insights

main 6 branches 26 tags

Go to file Code

About

http request/response parser for c

nodejs node

Readme MIT license Code of conduct Security policy 6.2k stars 347 watching 1.6k forks Report repository

Releases 26 tags

Packages No packages published

Contributors 81

Languages

File	Description	Time
contrib	Update license boilerplate.	6 years ago
fuzzers	Add another fuzzer for the urls.	3 years ago
.gitignore	header: treat Wine like MinGW	8 years ago
.mailmap	Bump version to 2.4.0	9 years ago
.travis.yml	send travis irc notifications to #node-cl	9 years ago
AUTHORS	src: support LINK/UNLINK (RFC 2068, draft-snell-link-method)	8 years ago
LICENSE-MIT	Update license boilerplate.	6 years ago
Makefile	v2.9.4	3 years ago
README.md	doc: add maintenance notice to readme	3 years ago
bench.c	bench: report throughput	5 years ago
http_parser.c	Allow Content-Length and Transfer-Encoding: chunked	3 years ago
http_parser.gyp	build: add default build flags to gyp script	11 years ago
http_parser.h	Allow Content-Length and Transfer-Encoding: chunked	3 years ago
test.c	Fix sizeof(http_parser) assert	3 years ago

README.md

## HTTP Parser

http-parser is **not actively maintained**. New projects and projects looking to migrate should consider [llhttp](#).

[Build Status](#)

This is a parser for HTTP messages written in C. It parses both requests and responses. The parser is designed to be used in performance HTTP applications. It does not make any syscalls nor allocations, it does not buffer data, it can be interrupted at anytime. Depending on your architecture, it only requires about 40 bytes of data per message stream (in a web server that is per connection).

Features:

C 96.6% Makefile 2.2% Python 1.2%

# http\_parser: the goods



## Good performance

The original parser had good performance, around 1.5 million requests per second.



## Lenient

It supported both HTTP/0.9, HTTP/1.0 and HTTP/1.1 specs and non compliant clients.



## Well tested

It had a comprehensive and battle tested test suite.

# http\_parser: the bads



## **Written in C**

It is not the most user friendly language.



## **Hard to maintain**

Over the course of ten years, the codebase became unmaintainable.



## **Vulnerability-prone**

The maintainability problem made it also hard to promptly address bugs and vulnerabilities.

# The current parser

**llhttp** is the current HTTP parser.

Written by Fedor Indutny in 2019, is the default since Node.js 12.

The screenshot shows the GitHub repository page for `nodejs/llhttp`. The repository is public and has 1.4k stars, 43 watchers, and 151 forks. It is currently on the `main` branch with 11 branches and 84 tags. The repository description is "Port of http\_parser to llparse".

The repository contains the following files and folders:

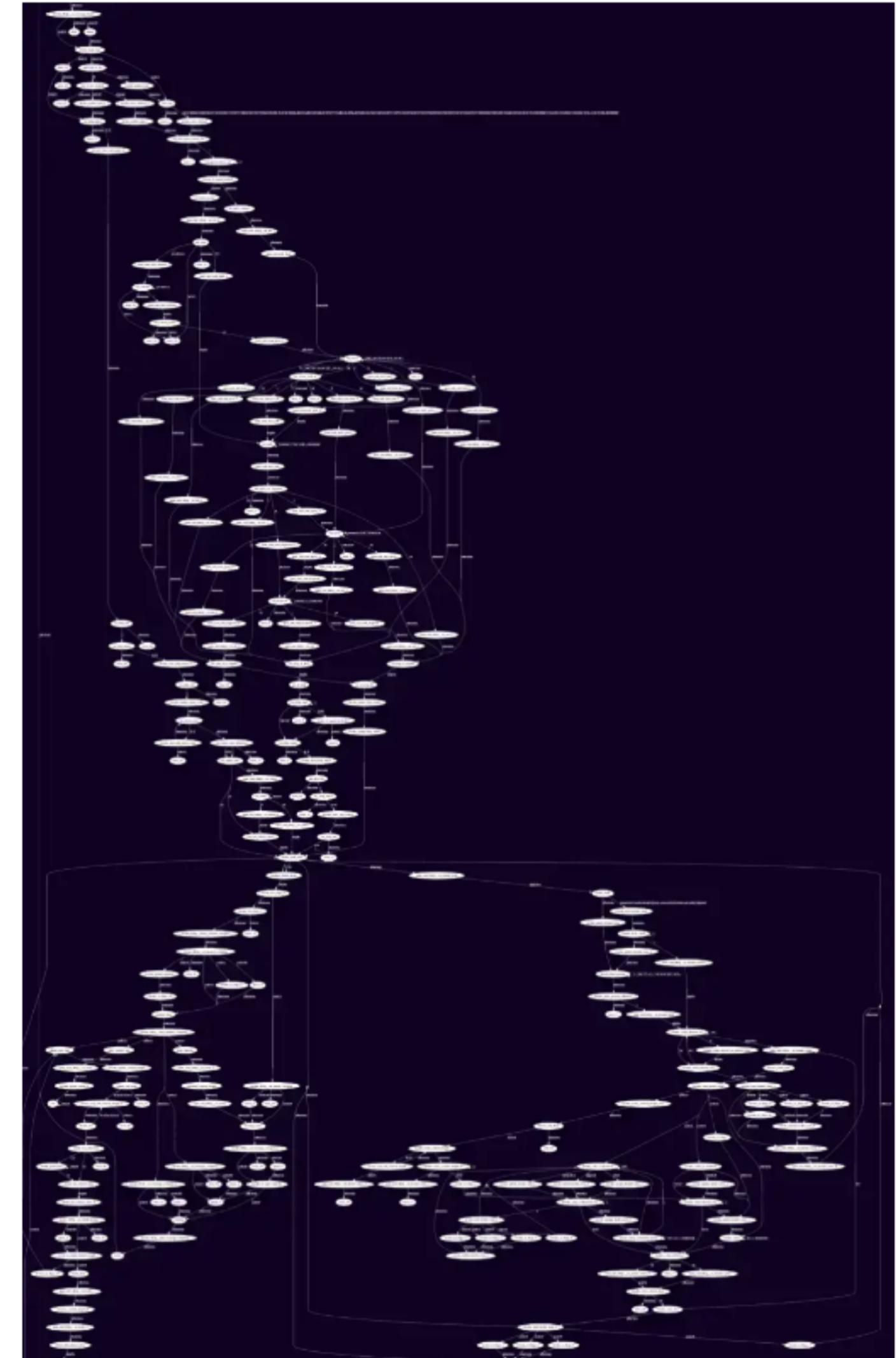
File/Folder	Description	Last Commit
<code>.github/workflows</code>	Upgrade GitHub Actions (#171)	9 months ago
<code>bench</code>	fix: prevent run bench without npm test (#209)	5 months ago
<code>bin</code>	feat: Allow to select WASM platform when using Docker. (#215)	4 months ago
<code>docs</code>	feat: Added release guide. (#173)	9 months ago
<code>examples</code>	src: wasm build support	2 years ago
<code>images</code>	images: add graphviz output	5 years ago
<code>src</code>	Stricter parsing of status line (#217)	3 months ago
<code>test</code>	Stricter parsing of status line (#217)	3 months ago
<code>.dockerignore</code>	src: wasm build support	2 years ago
<code>.eslintrc.js</code>	lib: update to llparse@2.0.0-beta9	6 years ago
<code>.gitignore</code>	make: re:release	5 years ago
<code>.npmrc</code>	enable package-lock	10 months ago
<code>CMakeLists.txt</code>	feat: Make release variables mandatory. (#194)	8 months ago
<code>CNAME</code>	Create CNAME	5 years ago
<code>CODE_OF_CONDUCT.md</code>	doc: move to main as primary branch (#130)	2 years ago
<code>Dockerfile</code>	feat: Allow to select WASM platform when using Docker. (#215)	4 months ago
<code>LICENSE-MIT</code>	gyp: changes	5 years ago
<code>Makefile</code>	chore: Fixed build script.	8 months ago
<code>README.md</code>	Update CMake docs (#221)	2 months ago
<code>_config.yml</code>	Set theme jekyll-theme-midnight	5 years ago
<code>libllhttp.pc.in</code>	enhance CMakeLists.txt	2 years ago
<code>package-lock.json</code>	build(deps): bump minimatch from 3.0.4 to 3.1.2 (#212)	4 months ago
<code>package.json</code>	feat: Allow to select WASM platform when using Docker. (#215)	4 months ago
<code>tsconfig.json</code>	http: reset header state on general header	5 years ago
<code>tslint.json</code>	src: port to TS	5 years ago

The repository also includes a `README.md` file. The right sidebar shows repository statistics: 1.4k stars, 43 watching, 151 forks, and 40 releases. The latest release is `v8.1.0` on Oct 11, 2022. There are 16 packages published, 16 users using the repository, 42 contributors, and 1 environment (github-pages) active.

# How does it work?

llhttp is a state based HTTP parser based on llparse.

llparse is capable to generate a very performant C code out of a TypeScript description of the possible states.





**Wait, what?!**

WHAT  
DO YOU  
MEAN  
?

**Yes, you  
got it right!**



# There's more!

The `http_parser` test suite has been ported and extended in `llhttp`.

The test suite is now described in **Markdown**, transpiled at runtime to a **C source code** and then compiled and executed.

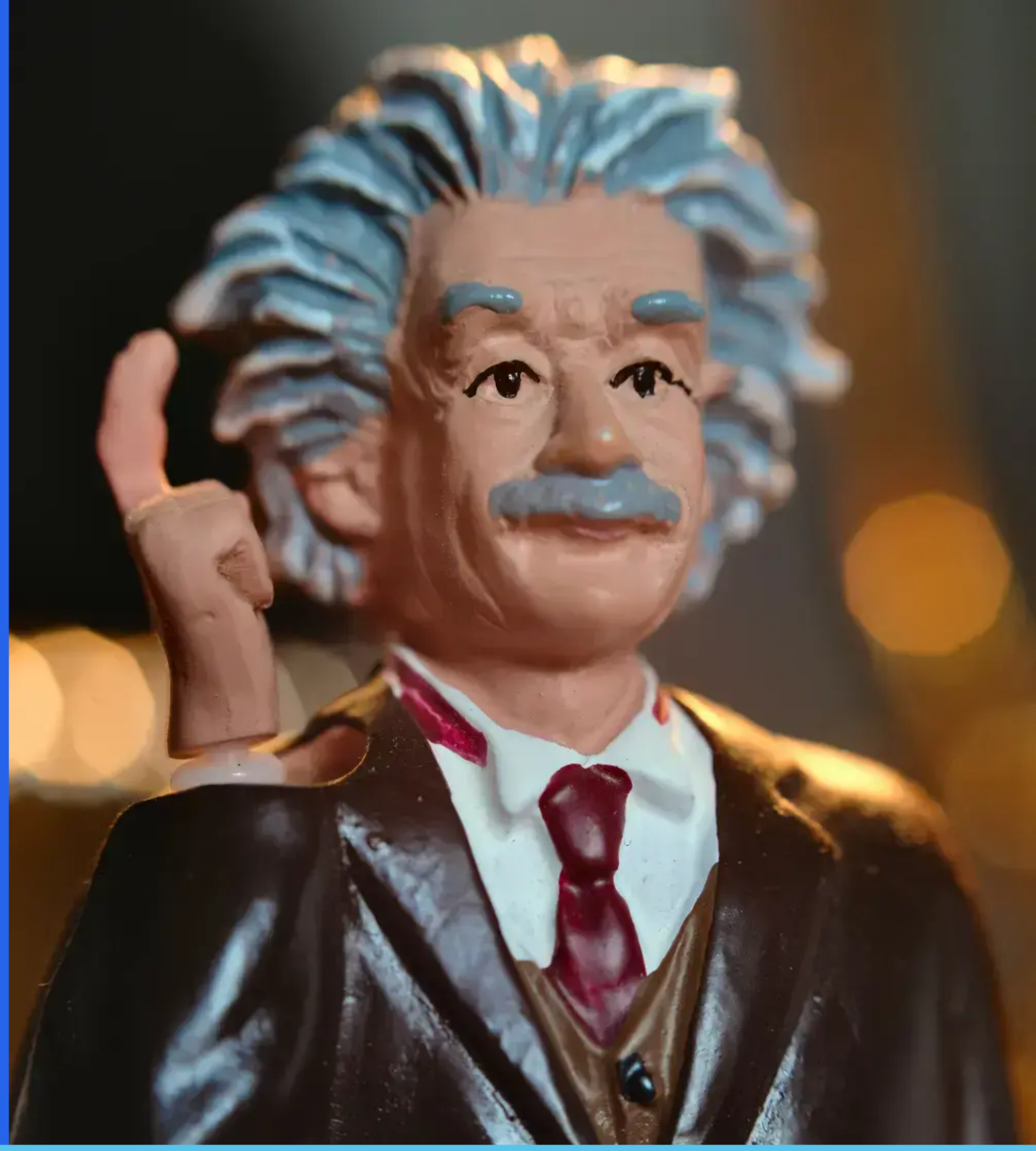
```
### Invalid HTTP version with lenient
```

```
```http
GET / HTTP/5.6
```

```
...
```

```
```log
off=0 message begin
off=0 len=3 span[method]="GET"
off=3 method complete
off=4 len=1 span[url]="/"
off=6 url complete
off=11 len=3 span[version]="5.6"
off=14 version complete
off=18 headers complete method=1 v=5/6 flags=0 content_length=0
off=18 message complete
```
```

**That's a genius  
in action!**



# llhttp: what's wrong with it?



## **Hard to debug and release**

The transpilation makes hard to debug issues.



## **Backward compatibility**

Supporting obsolete versions of HTTP introduces unneeded complexity.



## **You give them a finger, they take the arm**

Leniency-prone approach opens the door for a lot of edge cases.

**Where are  
the docs?**



**Do we have the  
solution?**



**Yes, start fresh!**





# Keep the goods



## **Piece of art design**

llhttp has a wonderful and performant architecture.



## **Testability**

The test suite is invaluable to ensure correctness.

# What shall we change?



## **Better transpilation**

Try to transpile to a human readable form and then generate static libs and include headers.



## **Use a higher level language**

Rust is a good candidate that can be used for both developer input and transpilation output.



## **Modern and strict**

Only support latest HTTP/1.1 standard (RFC 9110) and initially do have any leniency logic.

**Work in progress,  
stay tuned!**



# One last thing™

*“If there is no struggle,  
there is no progress.”*

**Frederick Douglass**





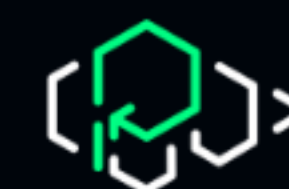
**Thank you!**

**Paolo Insogna**

**Node.js TSC, Principal Engineer**

@p\_insogna

paolo.insogna@platformatic.dev



**Platformatic**