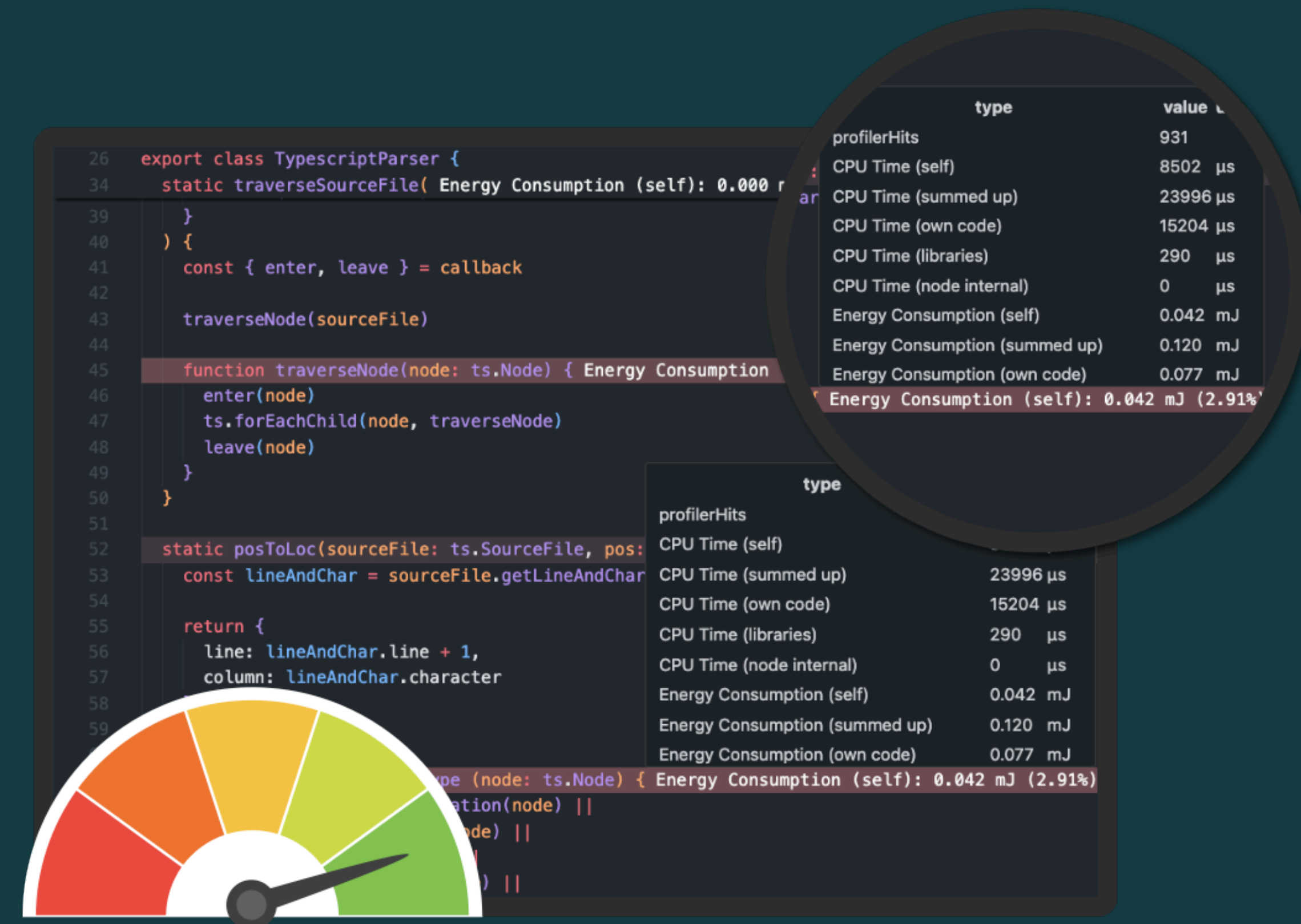




# Green your Code

Energy profiling for  
JavaScript & TypeScript

[www.oaklean.io](http://www.oaklean.io)

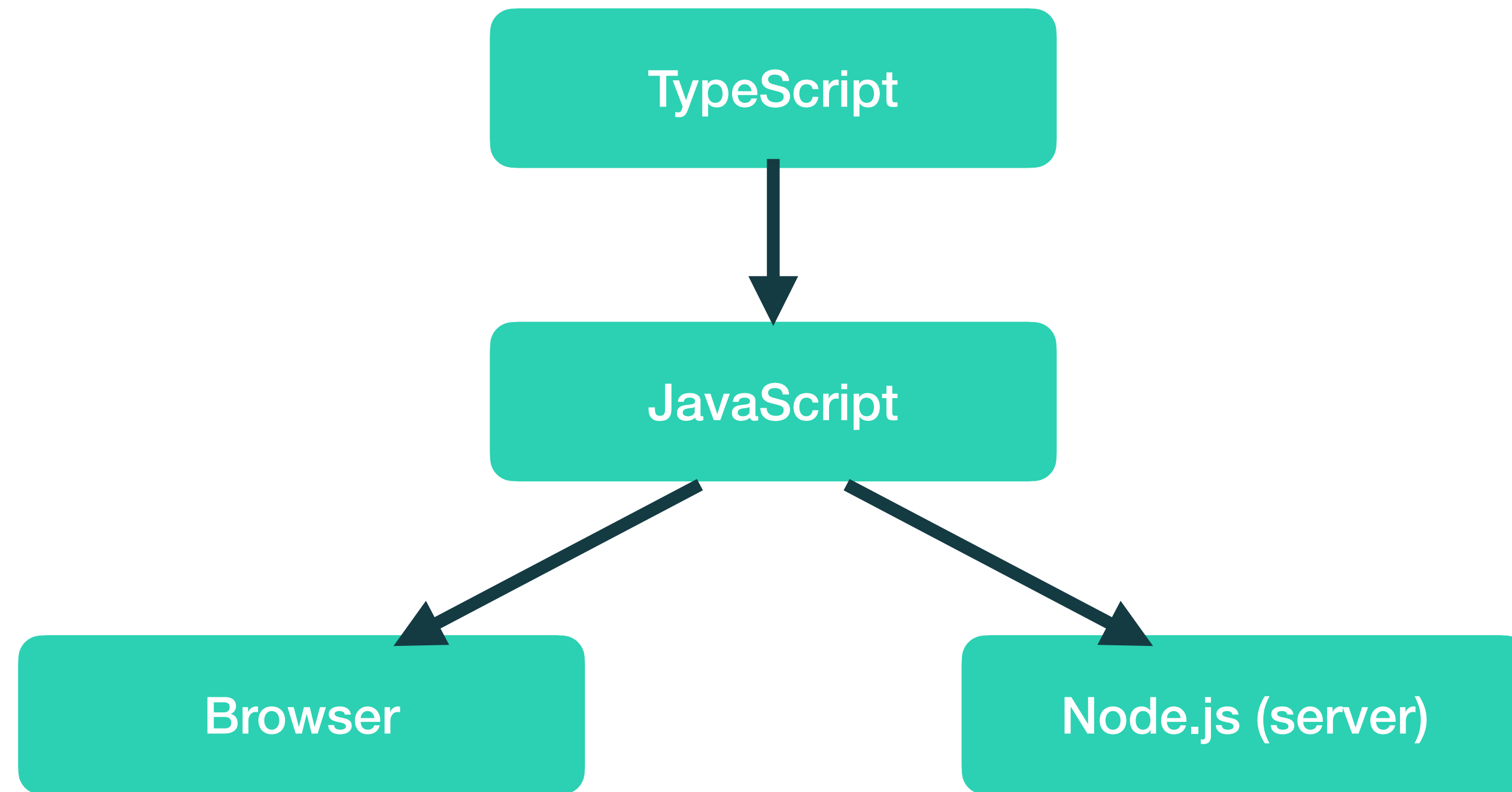


Open Source MIT License

# The Hidden Cost Of Every Line Of Code

- Developers usually focus on performance, readability, maintainability, but rarely on energy efficiency
- Every code choice (libraries, loops, APIs) affects energy footprint
- Existing tools don't measure energy in Node.js/JS/TS apps
- Oaklean: makes JS/TS energy usage visible and optimizable

# JavaScript/TypeScript/Node.js



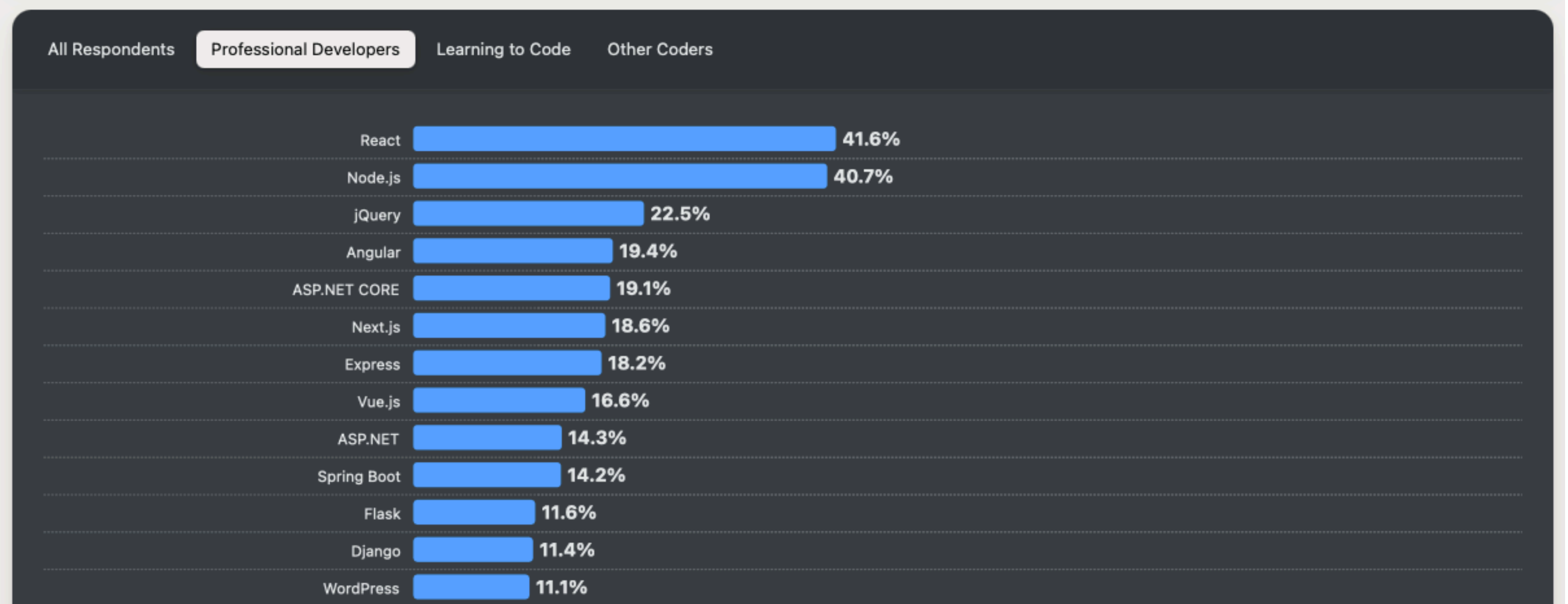
# Node.js At Scale = Energy At Scale

Why JavaScript/TypeScript Ecosystems Especially Need Sustainability Awareness

## Web frameworks and technologies

Node.js peaked in 2020 with its highest recorded usage score of 51%. While not as popular, it's still the most used web technology in the survey this year and has increased popularity among those learning to code from last year.

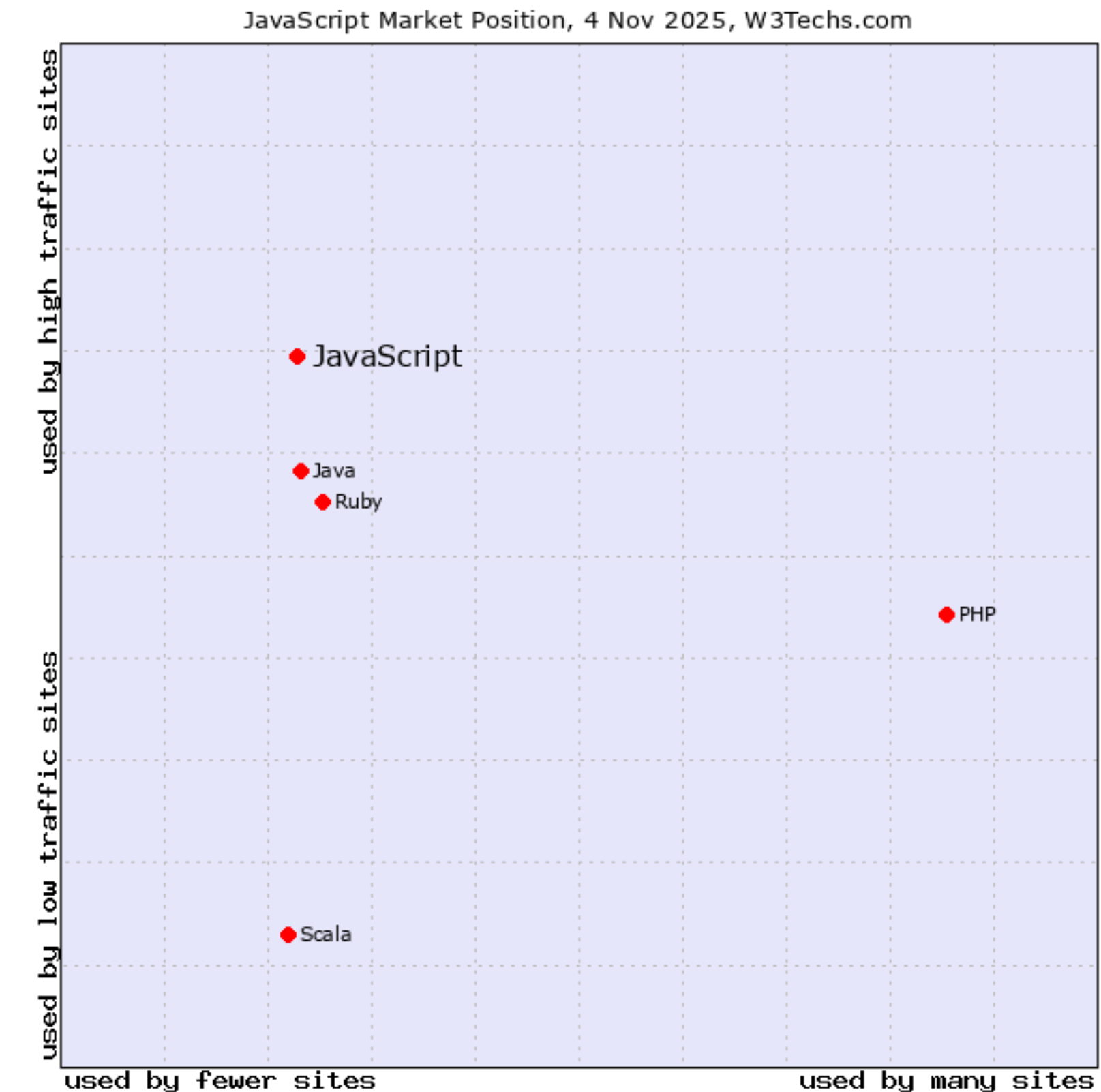
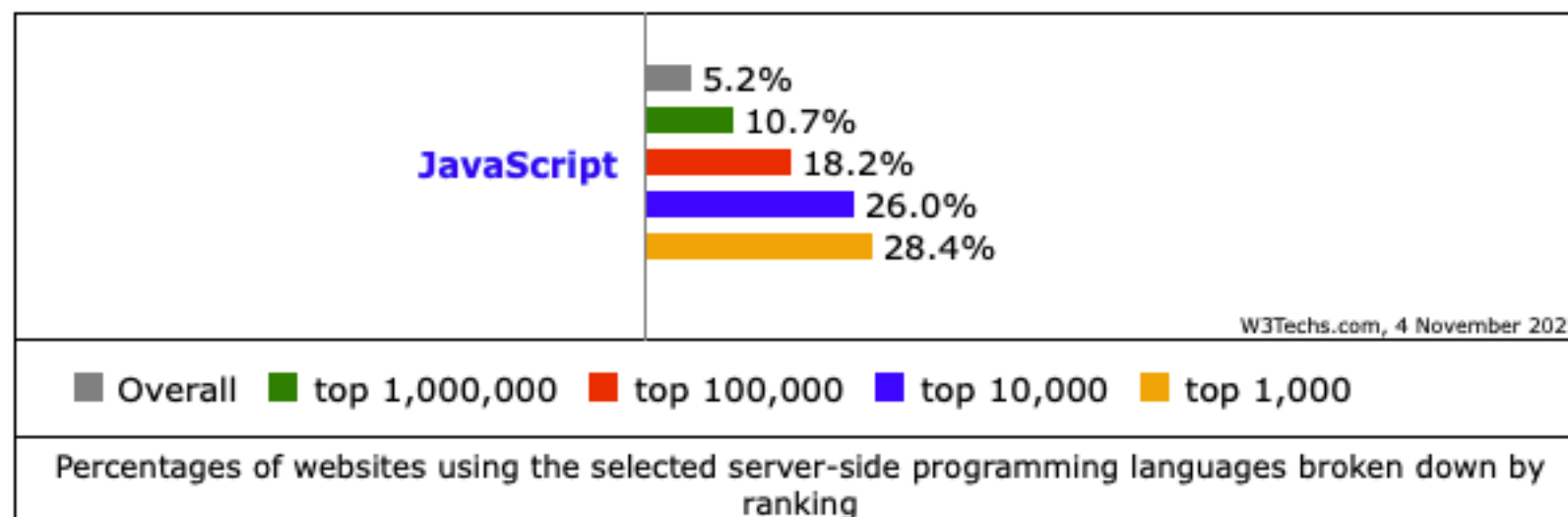
? Which **web frameworks and web technologies** have you done extensive development work in over the past year, and which do you want to work in over the next year? (If you both worked with the framework and want to continue to do so, please check both boxes in that row.)



# Node.js At Scale = Energy At Scale

## Why JavaScript/TypeScript Ecosystems Especially Need Sustainability Awareness

- Node.js used as website backends:
  - 5.2% of all websites
  - 10.7% by the top 1,000,000 websites
  - 28.4% by the top 1000 websites



# Node.js At Scale = Energy At Scale

## Why JavaScript/TypeScript Ecosystems Especially Need Sustainability Awareness

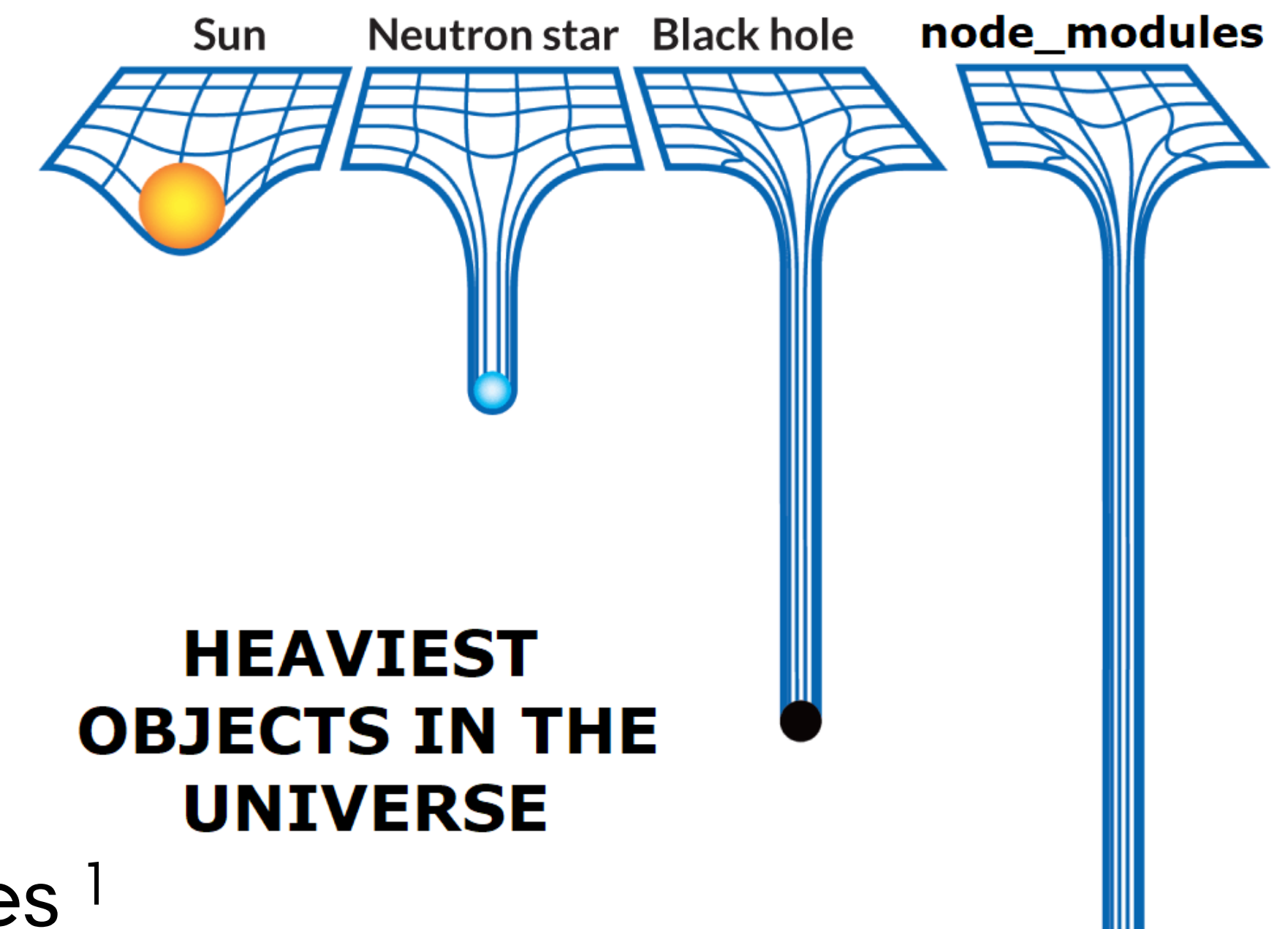
- **Paypal:** Adopted a unified JavaScript stack for frontend and backend
- **LinkedIn:** Migrated from Ruby on Rails to reduce overhead and server load
- **Netflix:** Server-side rendering and backend for global streaming
- **Uber:** Needed a high-concurrency backend to handle live GPS, surge pricing, and updates
- **Walmart:** Required event-driven architecture to handle traffic bursts
- **Trello:** Real-time UI updates for collaborative task management
- **eBay:** uses Node.js for parts of their large-scale marketplace backend



# Node.js At Scale = Energy At Scale

## Why JavaScript/TypeScript Ecosystems Especially Need Sustainability Awareness

- Dependencies = node\_modules
- 1.3 millions in January 2021
- 2017–2019: 700 new packages per day
- Many redundant packages:
  - 61% of the dependency functions were duplicates <sup>1</sup>
  - 10.4% were clones of other packages <sup>2</sup>
  - 17.92% are trivial (low/no functionality/only data) <sup>3</sup>



<sup>1</sup> <https://habr.com/en/articles/554334/>

<sup>2</sup> What the Fork? Finding Hidden Code Clones in npm

<sup>3</sup> <https://arxiv.org/abs/2510.04495>

# Meet Oaklean

## Measuring Energy In Your Codebase

- Open-source tool by Hitabis for JavaScript/TypeScript – Node.js
- Profiles CPU and memory energy consumption **per function or component**
- Reports energy cost directly in your workflow
- Empowers developers to code with environmental awareness



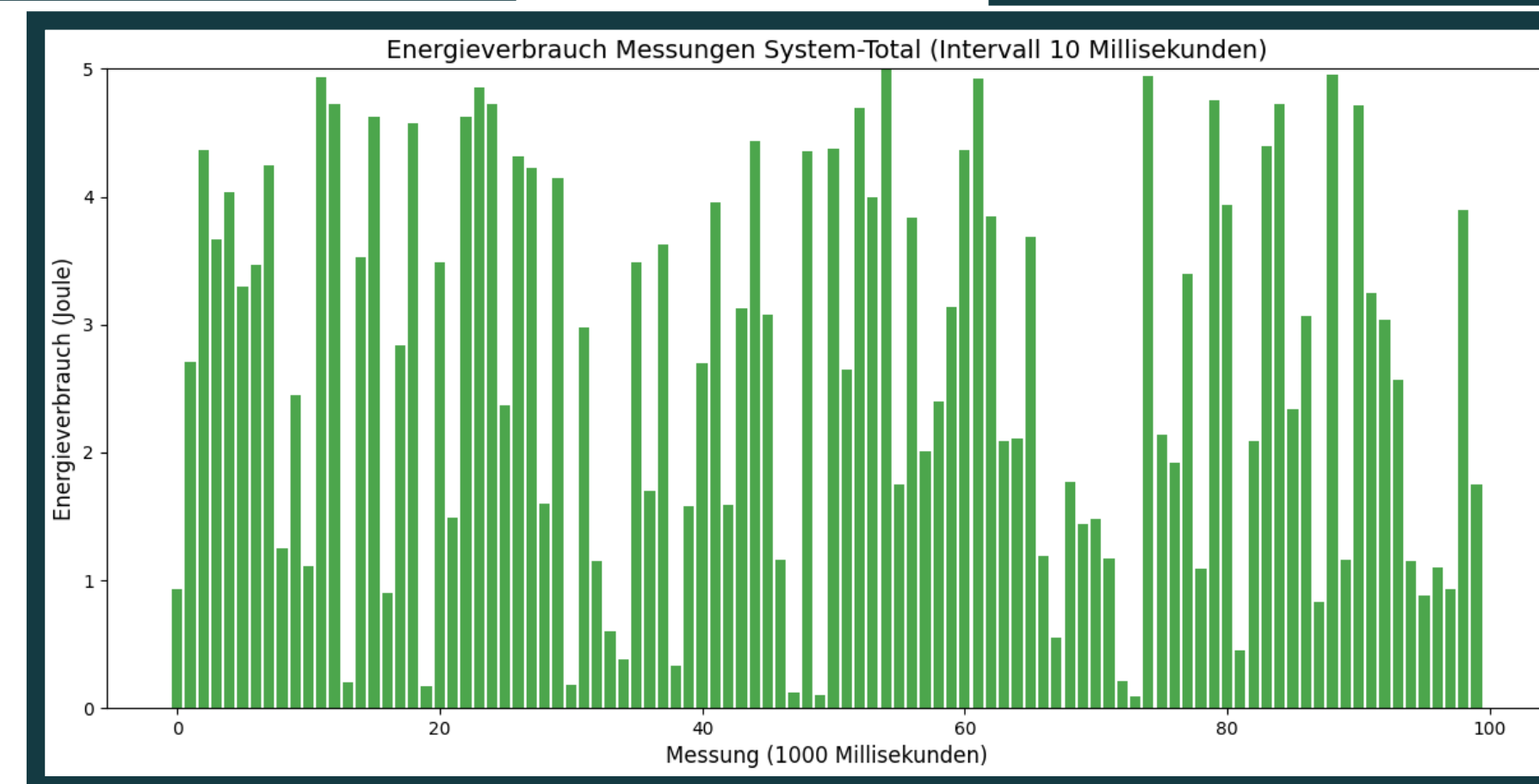
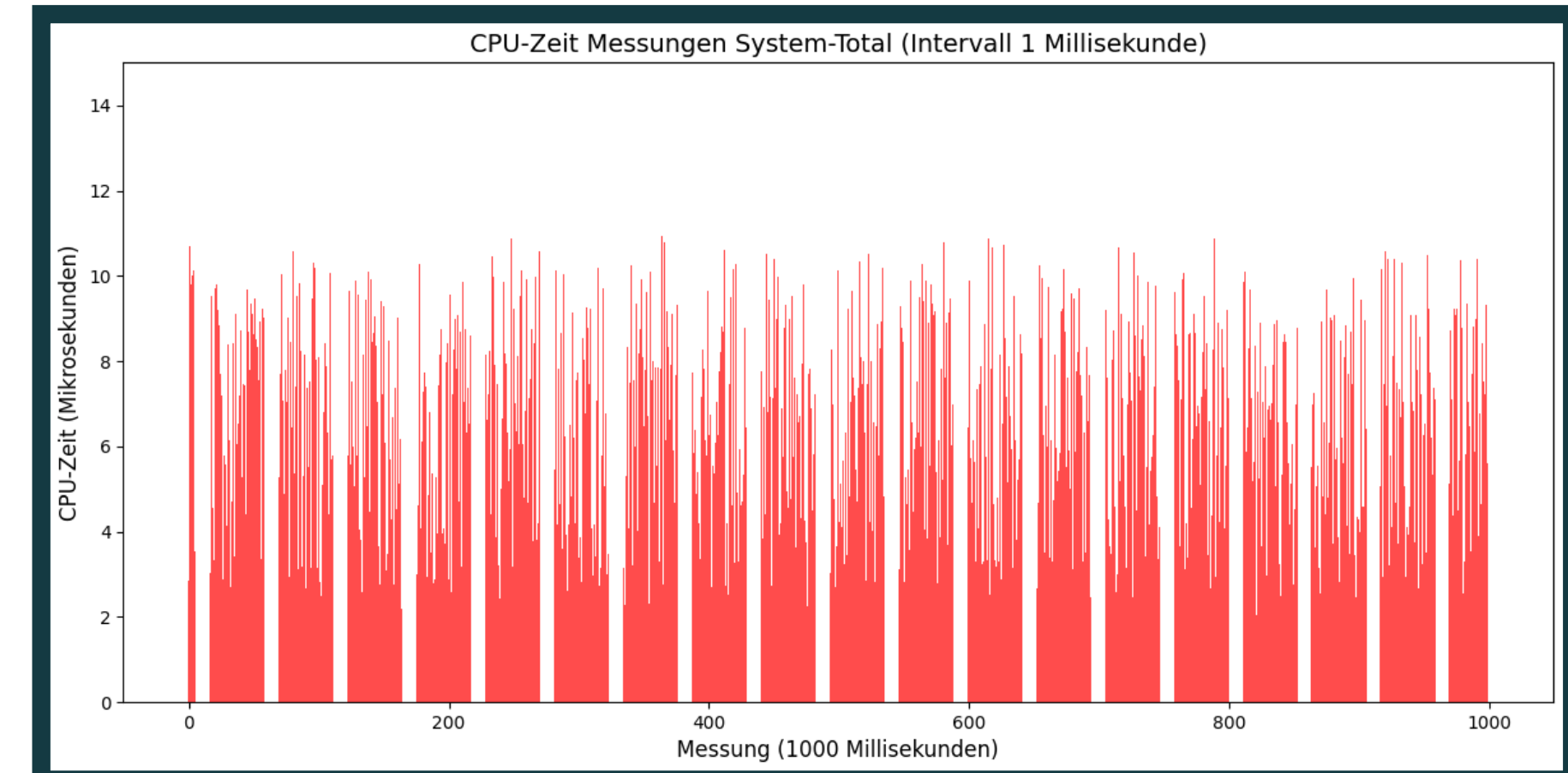
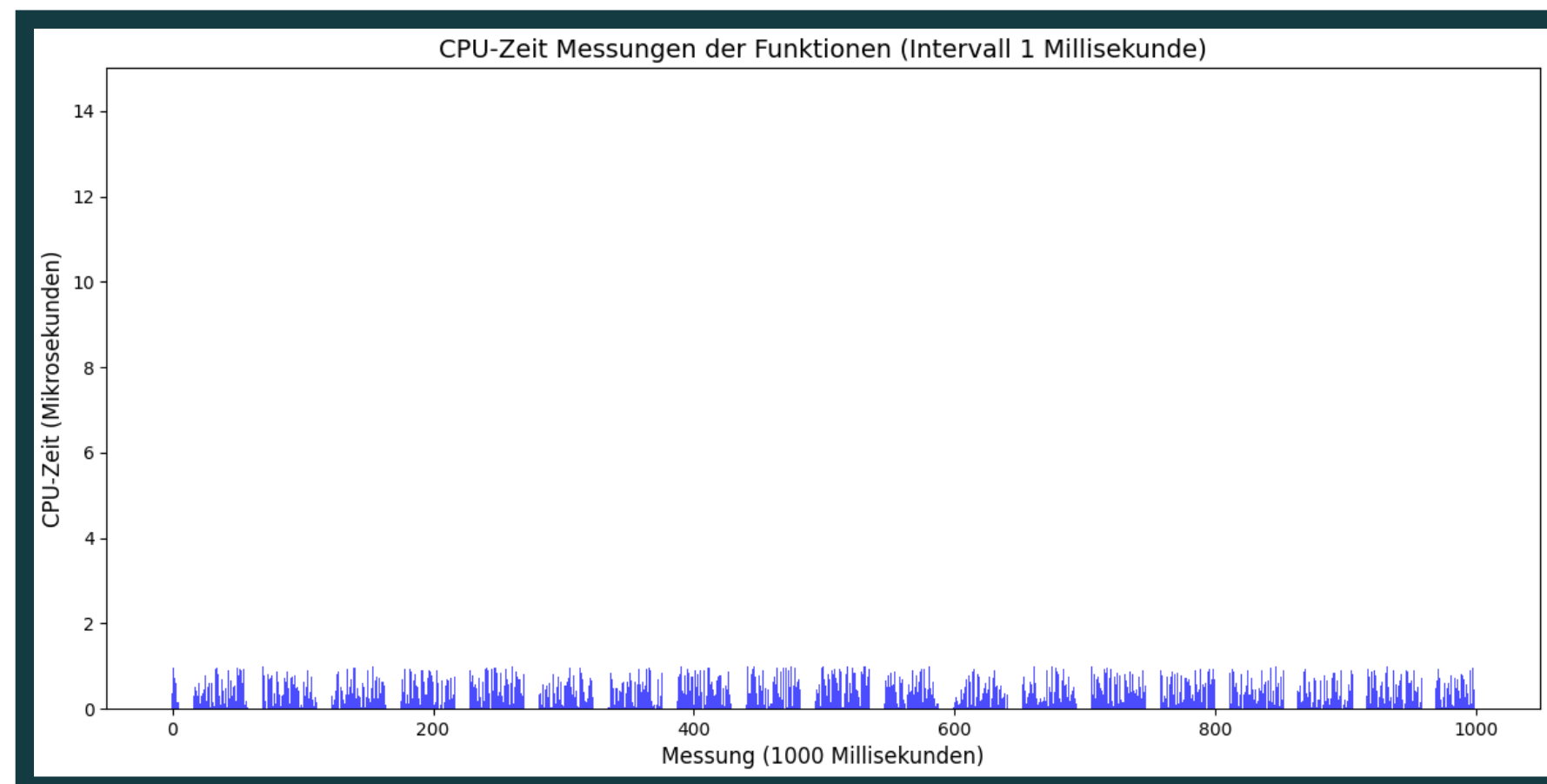




**Demo**

# How Oaklean Works Under The Hood

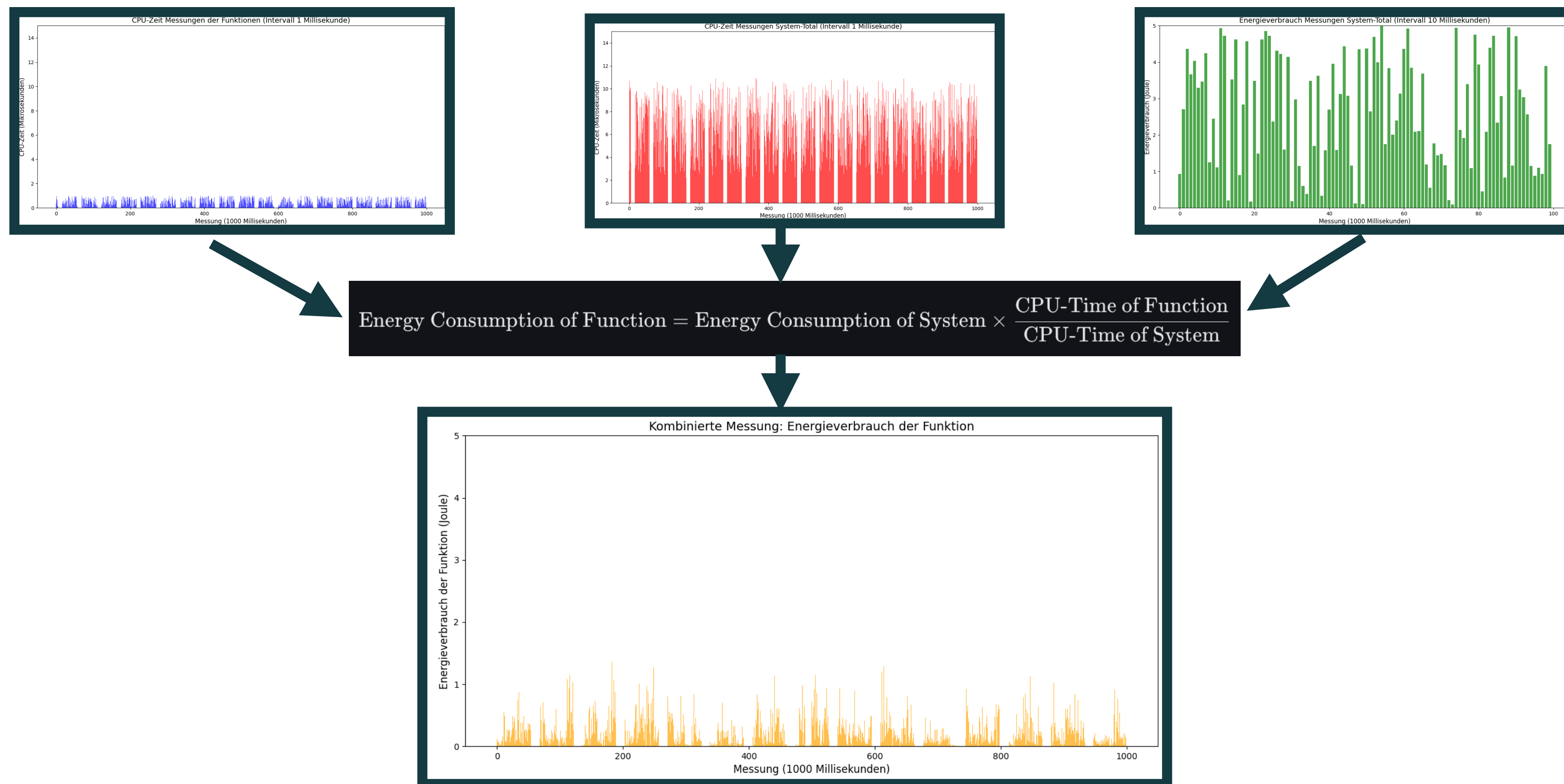
## Measuring Of CPU Time And Energy Consumption (CPU + RAM)



Collects low-level CPU and RAM energy data

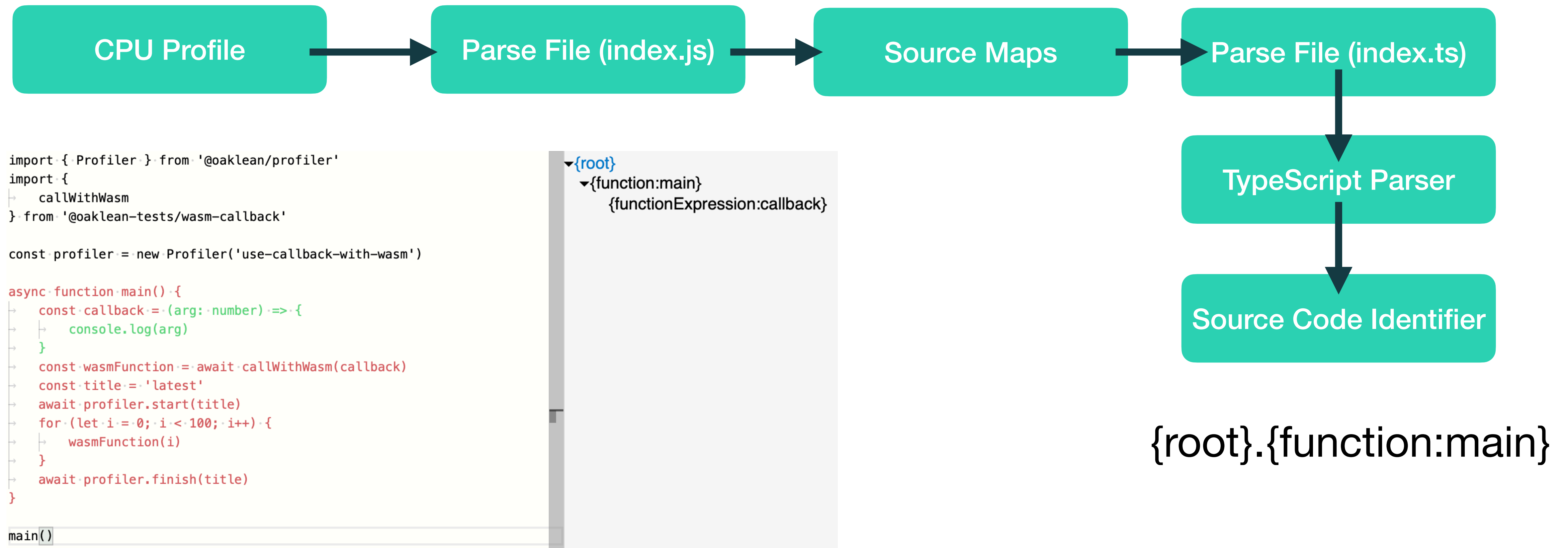
# How Oaklean Works Under The Hood

## Correlate Measurements



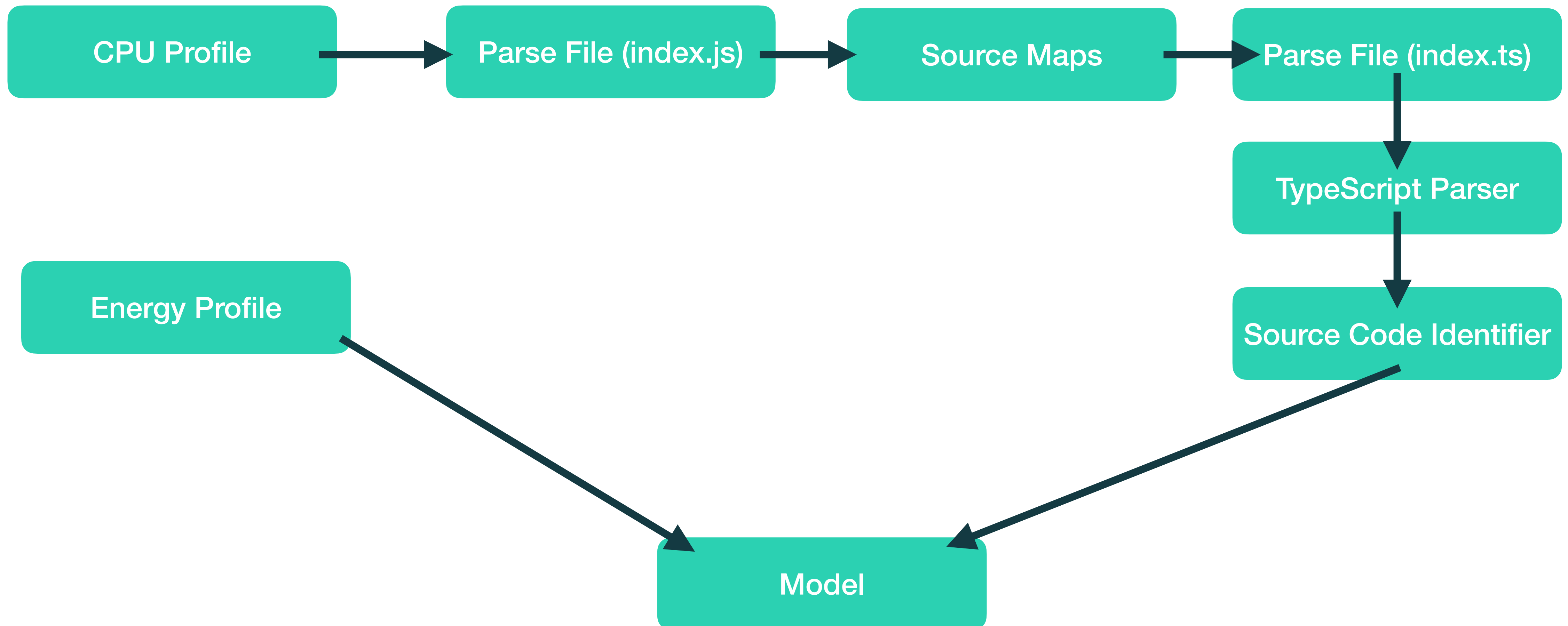
# How Oaklean Works Under The Hood

## Parsing The Source Code And Cluster Into Components



# How Oaklean Works Under The Hood

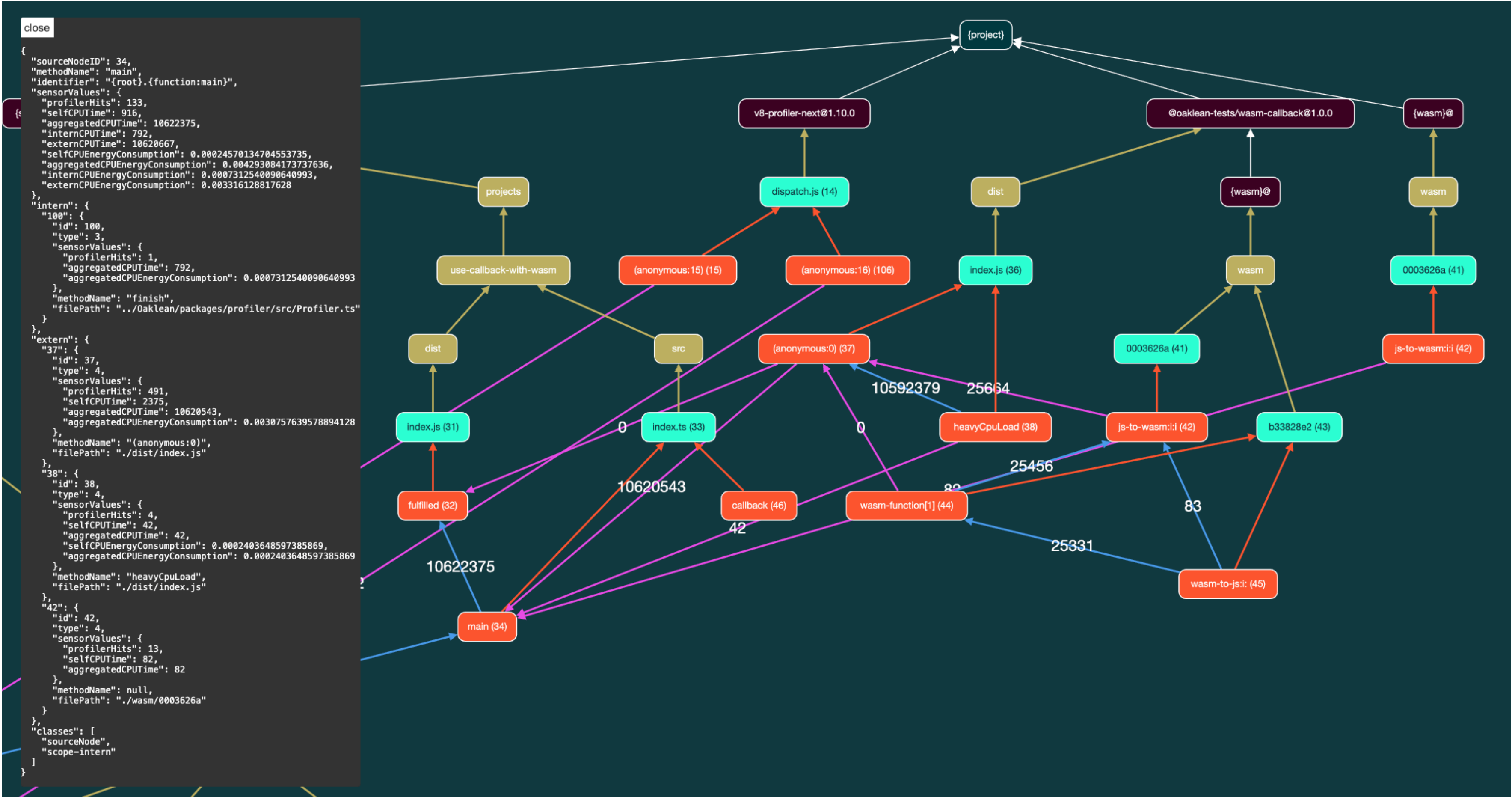
## Annotates Source Code With Cost Metrics





# How Oaklean Works Under The Hood

## Oaklean's Internal Model Representation



# How Oaklean Works Under The Hood

## Visualization In VS Code

OAKLEAN MEASUREMENTS

MEASUREMENTS

node\_modules 71.177 mJ

jest-runtime@29.5.0 30.322 mJ

build 30.322 mJ

index.js 30.322 mJ

node\_modules 26.453 mJ

jest-circus@29.5.0 25.825 mJ

jest-runner@29.5.0 9.118 mJ

typescript@5.9.2 4.645 mJ

expect@29.5.0 0.626 mJ

zod@3.24.2 0.150 mJ

emittery@0.13.1 0.116 mJ

ts-jest@29.1.2 0.106 mJ

@jest/transform@29.5.0 0.076 mJ

v8-profiler-next@1.10.0 0.070 mJ

@jest/reporters@29.5.0 0.053 mJ

cli-color@2.0.4 0.024 mJ

METHODS

Energy Consumption (summed up)

createDirectoriesRecursively 0.004 mJ

ExternalResourceHelper.ts

ExternalResourceHelper

listen 0.002 mJ

(anonymous:0) 5.725 mJ

fileInfoFromScriptID 4.875 mJ

(anonymous:0) 4.050 mJ

(anonymous:0) 0.207 mJ

isFrozen 0.014 mJ

loadFromFile 0.012 mJ

fromJSON 0.001 mJ

constructor 0.002 mJ

scriptIdS 0 mJ

sourceMapFromScriptID 0.004 mJ

replaceSourceMapByScriptID 0.001 mJ

sourceCodeFromScriptID 0.002 mJ

loadedFilePaths 0 mJ

JS index.js

node\_modules > jest-runtime > build > JS index.js > Runtime > getExportsOfCjs > reexports.forEach() callback

222 class Runtime {

779 getExportsOfCjs(modulePath) : any {

788 reexports.forEach(reexport => {

793 }

794 } else {

795 const resolved = this.\_resolveCjsModule(modulePath, reexport);

796 const exports = this.getExportsOfCjs(resolved);

797 exports.forEach(namedExports.add, namedExports);

798 }

799 });

800 this.\_cjsNamedExports.set(modulePath, namedExports);

801 return namedExports;

802 }

803 requireModule Energy Consumption (summed up): 27.713 mJ (36.32%)

804 (from, moduleName, options, isRequireActual = false) : any {

805 const isInternal = options?.isInternalModule ?? false;

806 const moduleID = this.\_resolver.getModuleID(

807 this.\_virtualMocks,

808 from,

809 moduleName,

810 {

811 conditions: this.cjsConditions

812 }

813 });

814 let modulePath;

815 }

PROBLEMS

OUTPUT

TERMINAL

GITLENS

SPELL CHECKER

MEASUREMENTS

DEBUG CONSOLE

PORTS

Energy Consumption (summed up)

Runtime

requireModuleOrMock 27.987 mJ

requireModule 27.713 mJ

\_loadModule 27.291 mJ

\_execModule 27.178 mJ

transformFile 19.278 mJ

readFile 0.860 mJ

readFileBuffer 0.753 mJ

\_shouldMockCjs 0.397 mJ

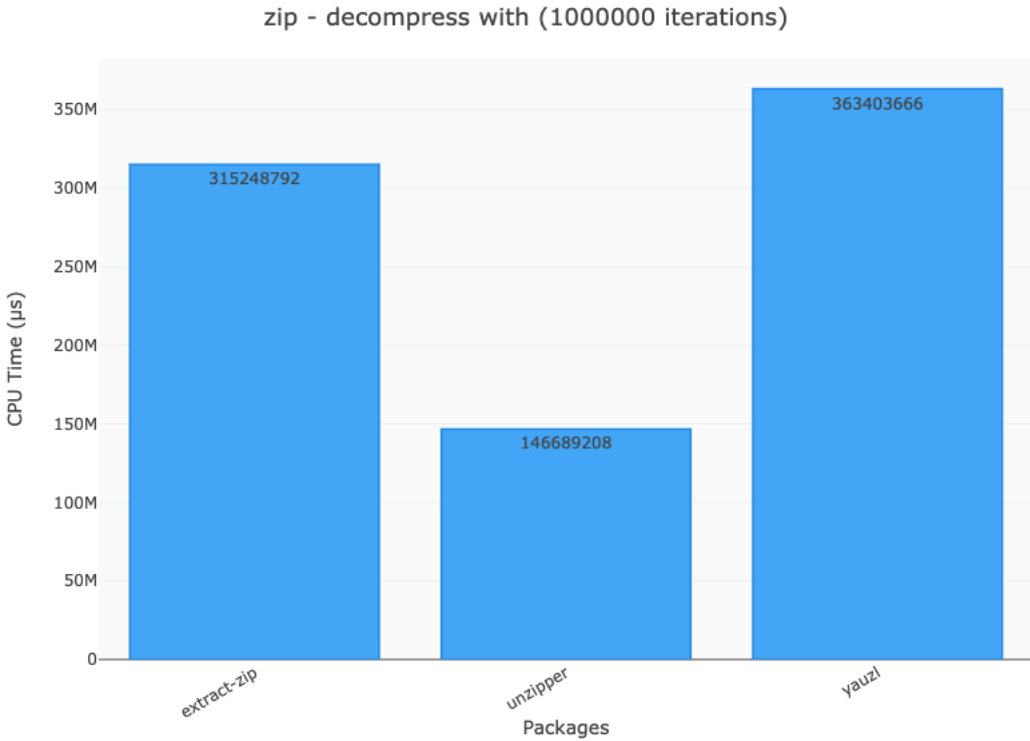
\_resolveCjsModule 0.257 mJ

# Case Study

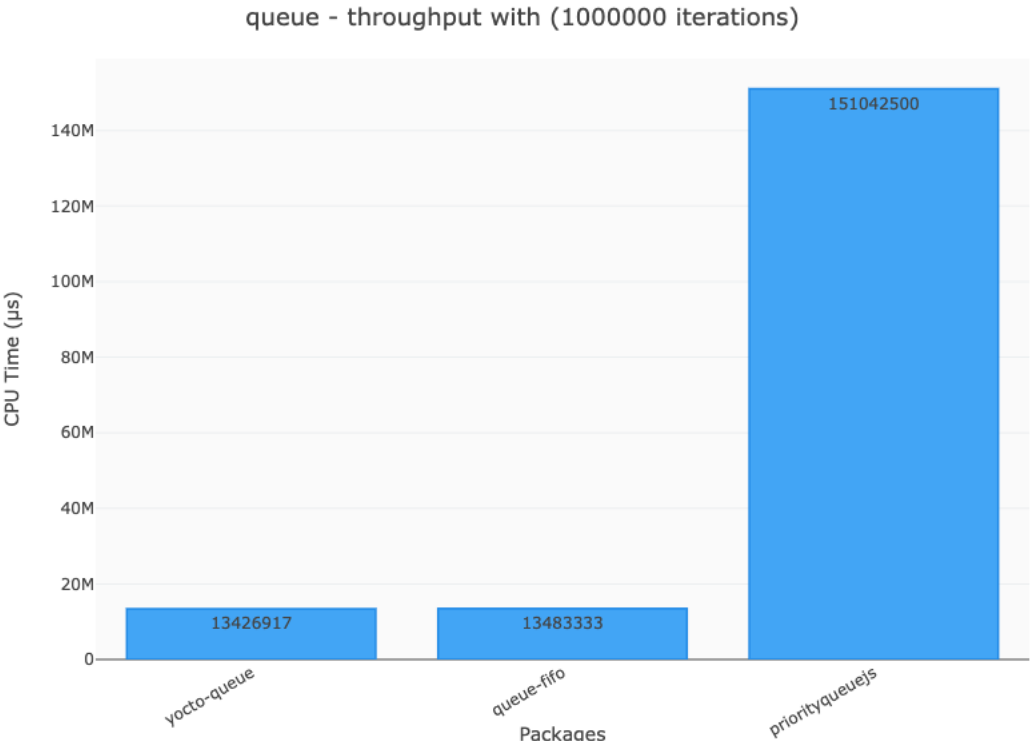
## Node Module Comparison

cpu time

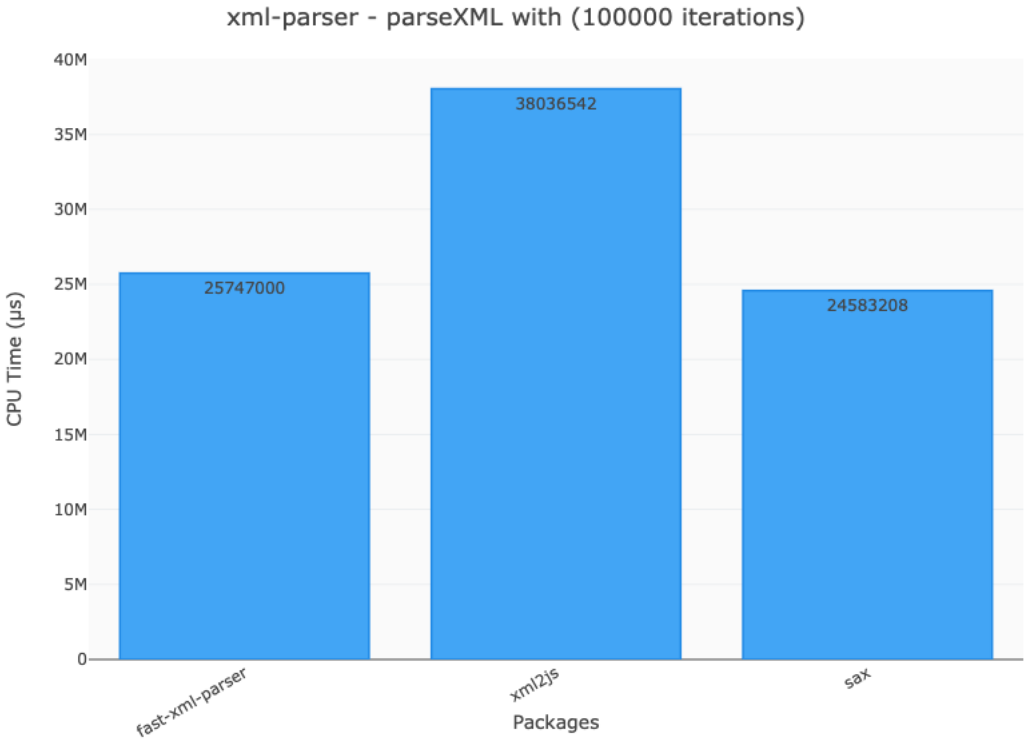
zip-decompress



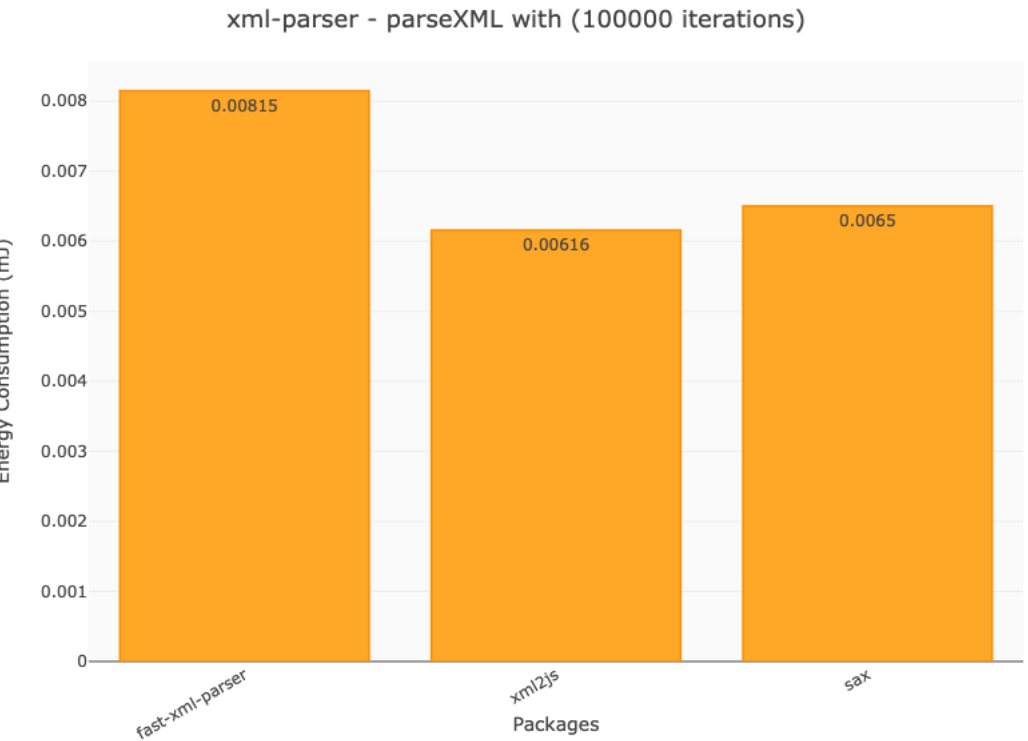
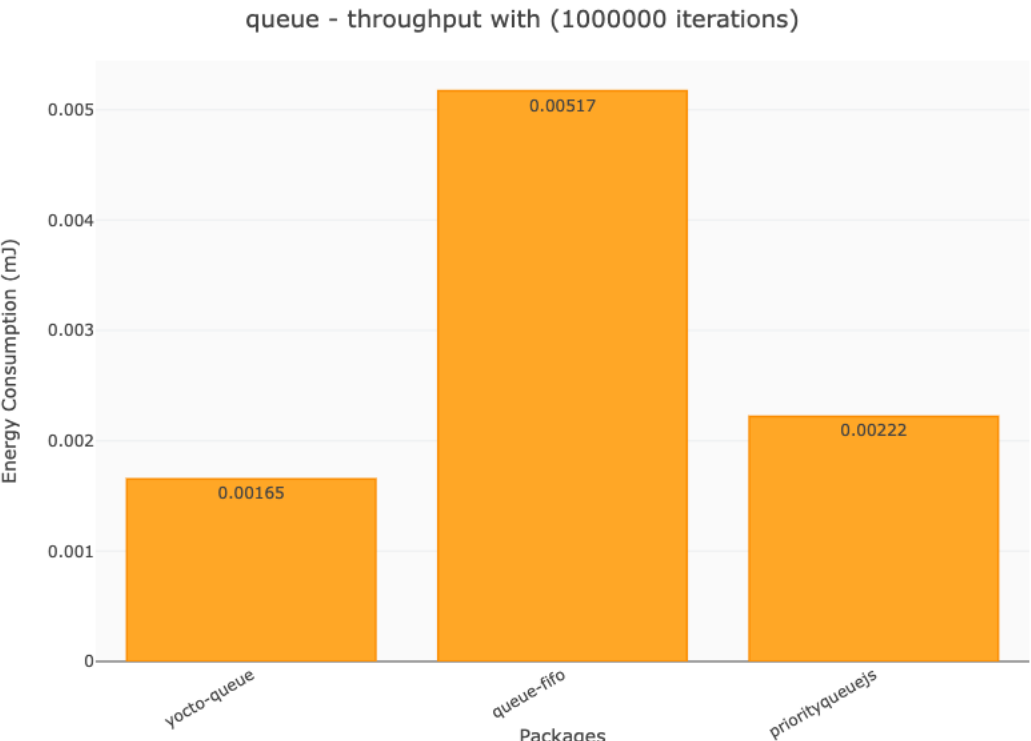
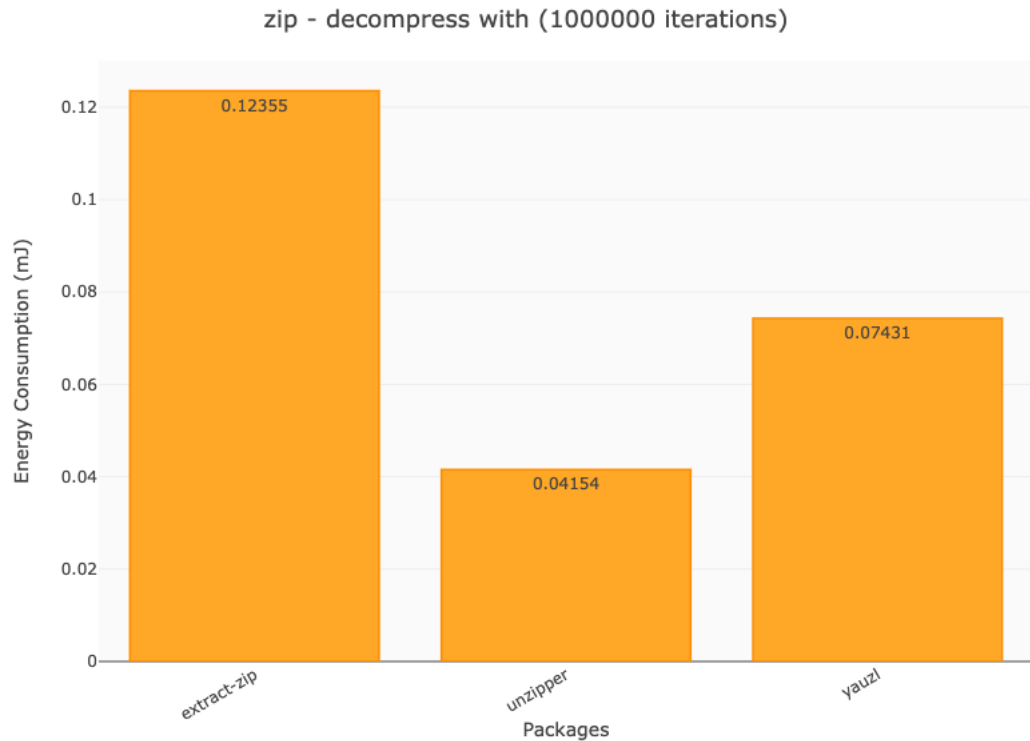
queue-throughput



xml-parser



energy consumption



# A Community Vision

## The Green Library Database

- Next step: build an open database of library footprints
- Share benchmarks: “How green is your dependency?”
- Encourage community contributions and comparisons
- In editor suggestions for „greener“ dependencies
- Contribute on GitHub: tools, metrics, libraries



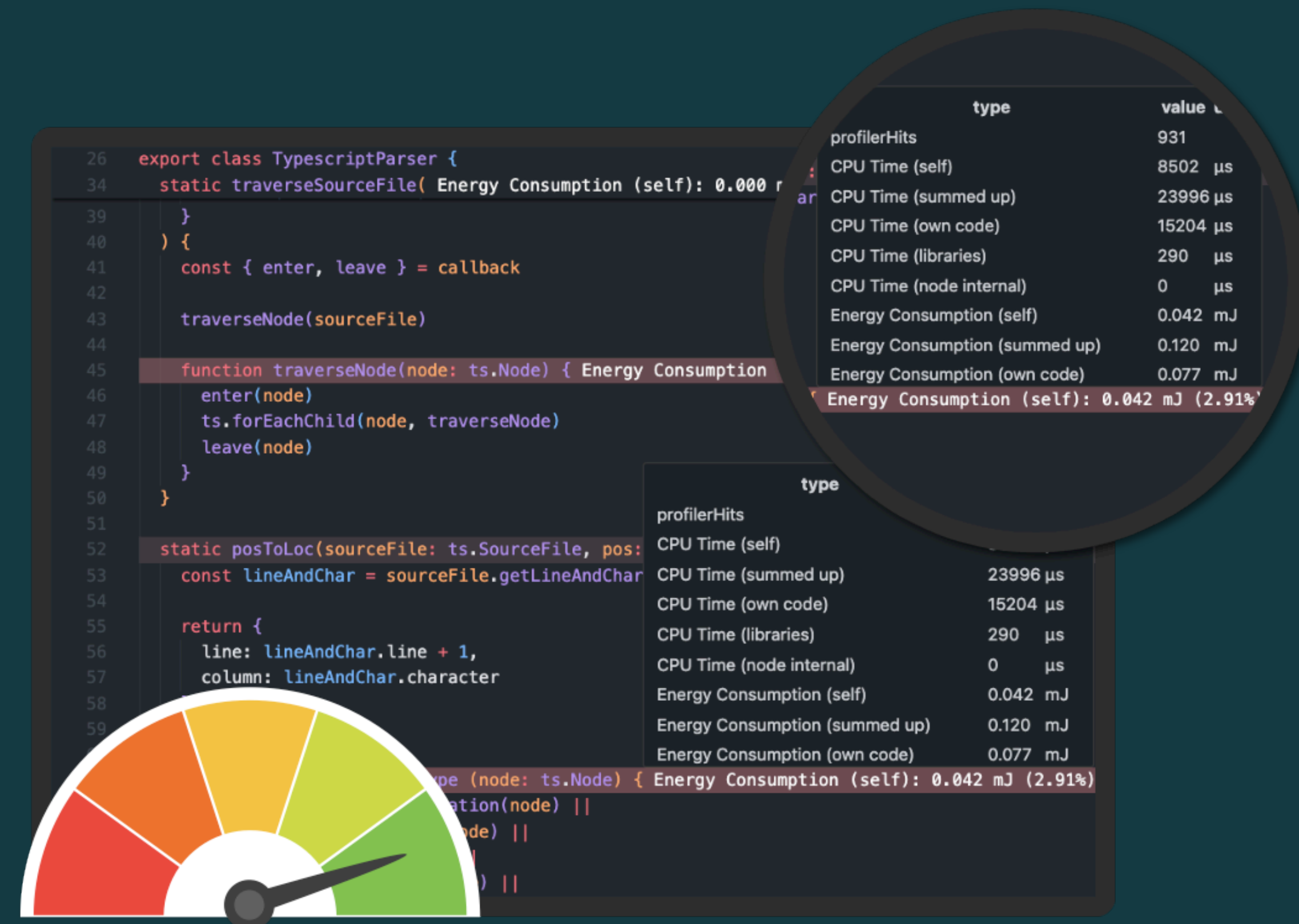


# Green your Code

[www.oaklean.io](http://www.oaklean.io)

Supports: Linux/Mac/Windows

Open Source MIT License





# Questions & Reflections

# Sources

- Stack overflow developer survey  
<https://survey.stackoverflow.co/2024/technology#most-popular-technologies-webframe-prof>
- Comparison of the usage statistics of JavaScript for websites  
<https://w3techs.com/technologies/comparison/pl-js>
- Why Top Companies Are Using Nodejs As A Backend  
<https://enstacked.com/why-top-companies-are-using-nodejs-for-backend>
- Top 10 Large Companies Using Node.js for Backend  
<https://medium.com/devmap/top-10-large-companies-using-node-js-for-backend-f32aa3e55cdd>
- „What the Fork? Finding Hidden Code Clones in npm“  
<https://ldklab.github.io/assets/papers/icse22-shrinkwrap.pdf>
- Only 39% of the functions in node\_modules are unique in the default Angular project  
<https://habr.com/en/articles/554334/>
- Detecting and Characterizing Low and No Functionality Packages in the NPM Ecosystem  
<https://arxiv.org/abs/2510.04495>