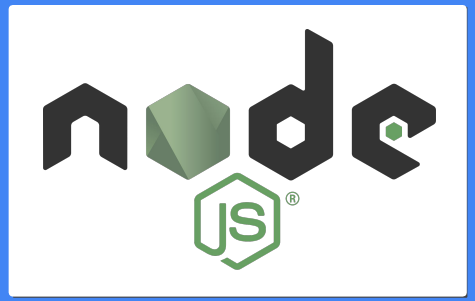
Introducing  in 

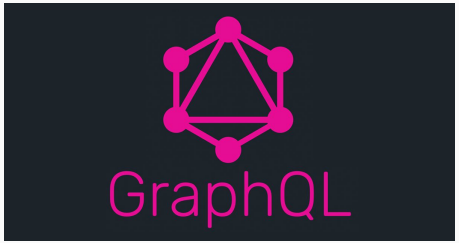
Ivar Conradi Østhus
@ivarconr

Trygve Lie
@trygve_lie

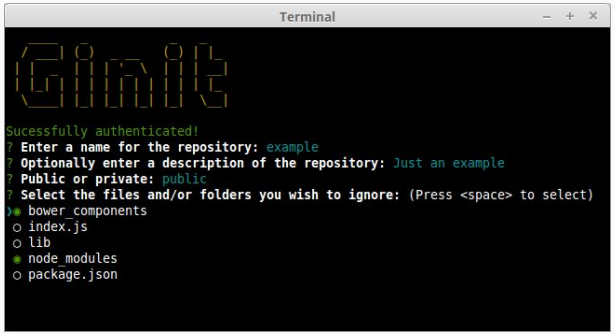
Agenda

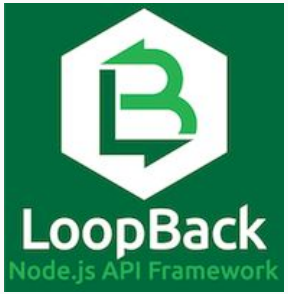
- ➔ What is Node.js?
- ➔ Why Node.js?
- ➔ How did we Introduce Node.js in FINN?
- ➔ Standardizing Node.js

Where is Node.js used?



express





What is Node.js?

- Server-Side JavaScript
- Built on Google’s V8
- Created by Ryan Dahl in 2009
- First version 2011
- Written i C, C++ and JavaScript

Non-blocking I/O

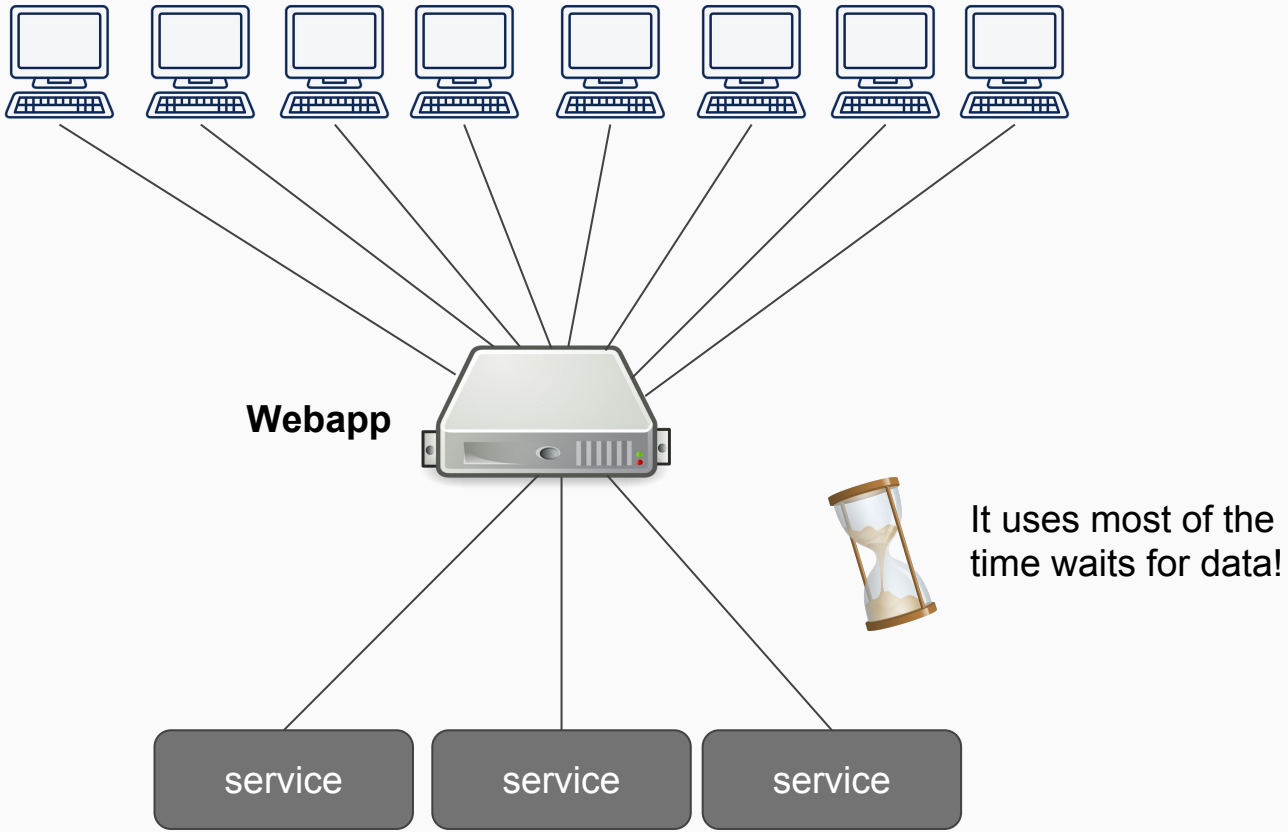
Blocking I/O

```
1.  const result = db.query('select x from table_y');
2.  doSomethingWithResult(result);
3.  doSomethingWithOutResult();
```

Non-blocking I/O

```
1.  const result = db.query('select x from table_y', (result) => {
2.    doSomethingWithResult(result);
3.  });
4.  doSomethingWithOutResult();
```

What do a webapp typically do?



It uses most of the time waits for data!

Threads are expensive!

Threads have significant overhead

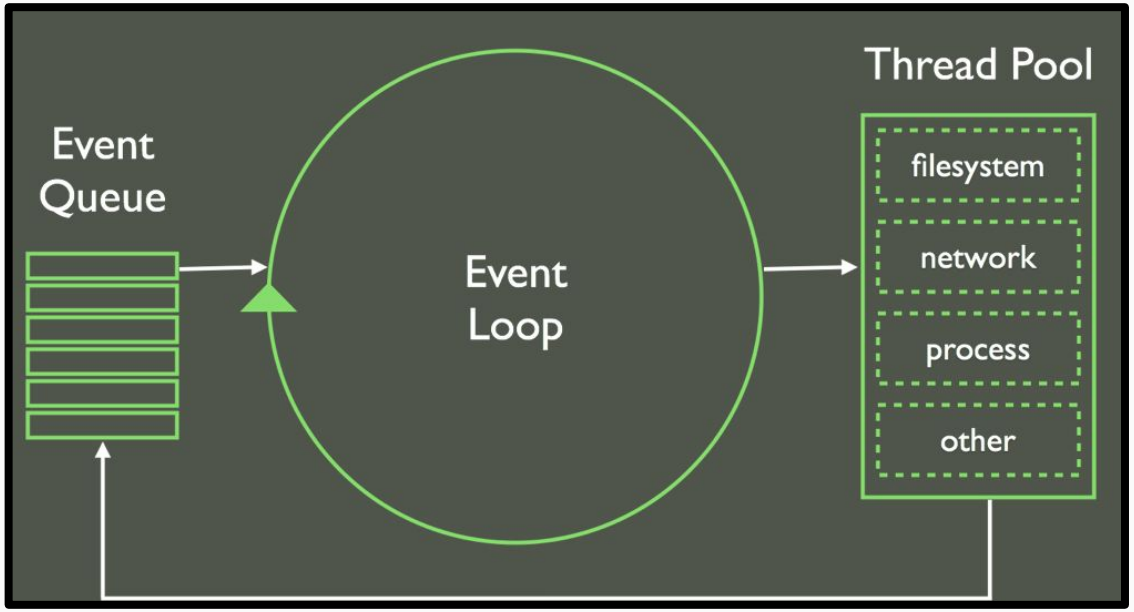
- Context switches
- Memory footprint
- CPU cycles

Why waste resources on **waiting**?

Node.js event-driven architecture

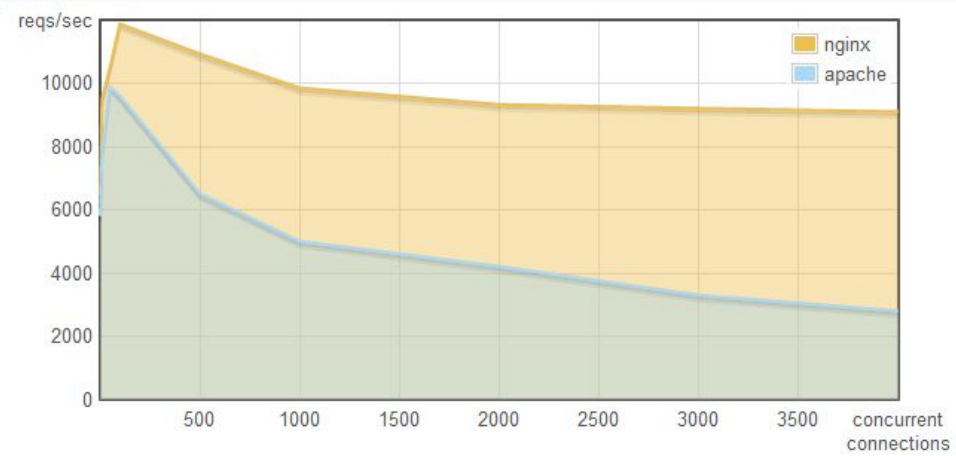
The main **event loop**

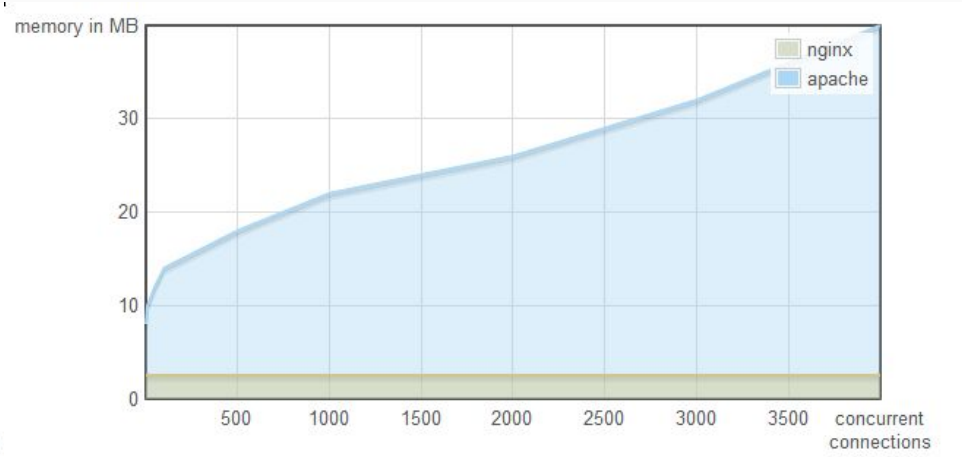
- “Single threaded”
- Non-blocking I/O



Handles thousands of concurrent connections with minimal overhead (CPU/Memory) on a single process

Threads vs. Event loop





https://blog.webfaction.com/2008/12/a-little-holiday-present-10000-reqssec-with-nginx-2/

Why Node.js?

- Already use JavaScript in the browser
- Mental switching
- Simplicity
- Modularity
- Scaling Node.js

We already use Node.js!

JavaScript in the browser

Tools build upon the Node.js ecosystem

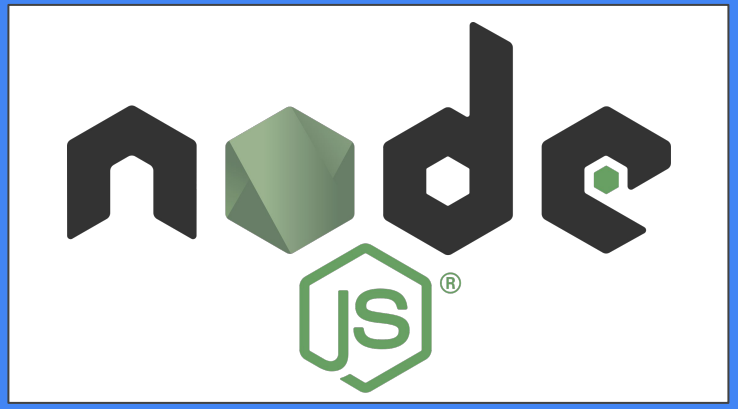
Fewer mental context switches

- Client-side and server-side in same language
- Possible to reuse code
- Learning Node.js is easy
  - **Learning JavaScript is the hard part!**

Browser

!==

Node.js



Simplicity!

- Fast start-up time, typically less than **1 second**.
- **We don’t deal with threads in our code!**
- JSON (JavaScript Object Notation) support built in!
- Great conventions
  - `npm install`
  - `npm run start`
  - `npm run test`
- Few abstractions, close to “web”!

Setting up a web server in Node.js

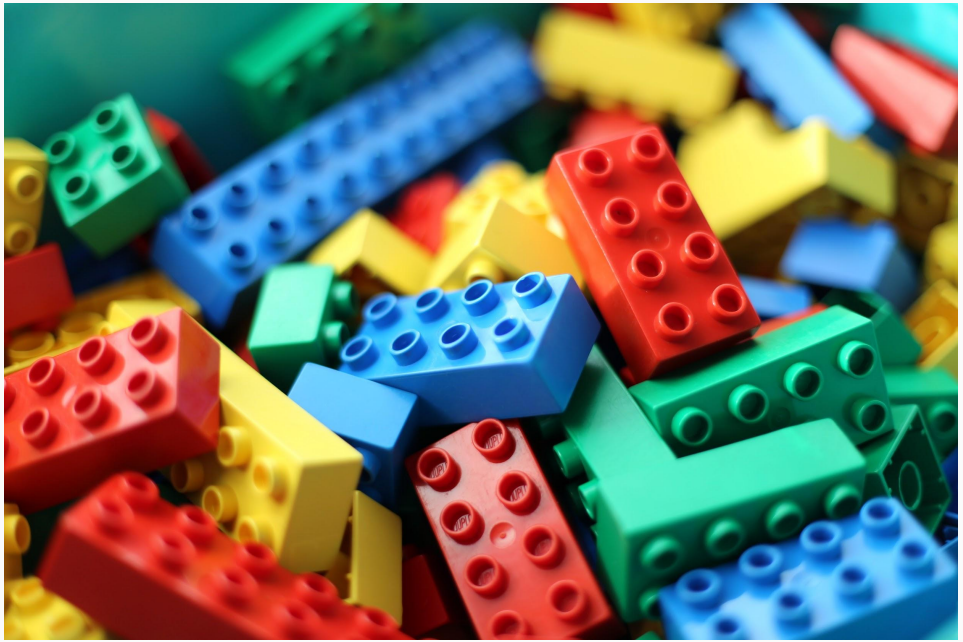
```
1.  const http = require('http');
2.
3.  const server = http.createServer((req, res) => {
4.    res.statusCode = 200;
5.    res.setHeader('Content-Type', 'text/plain');
6.    res.end('Hello World\n');
7.  }).listen(3000);
```

Node.js has a modular ecosystem!

Core Modules

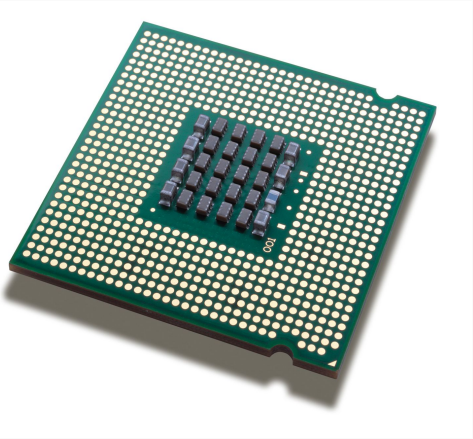
File Modules

Packages



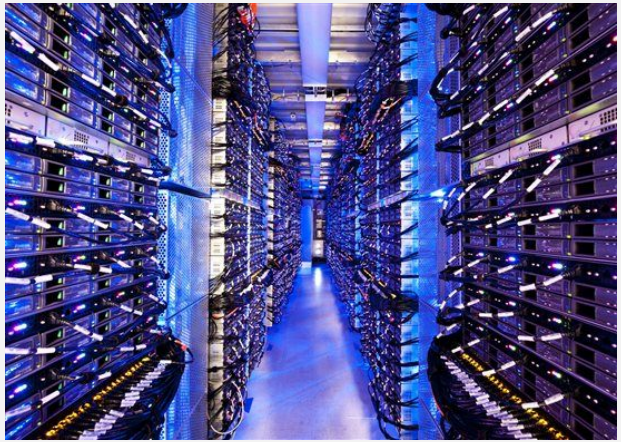
Scaling Node.js!

1. Multiple cores

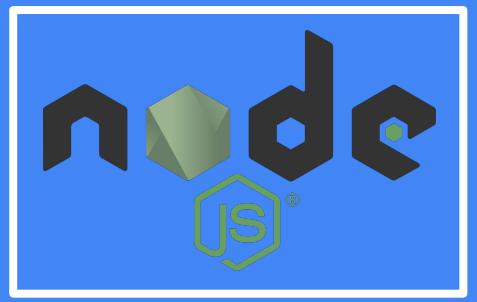


2. Multiple servers

3. Perfect for cloud!



AWS

How did we introduce  ?

Step 1: The Trojan Horse!

Started using Node.js to process frontend resources.

frontend-maven-plugin

- Downloads & installs Node and NPM locally
- Correct Node & npm versions in all build environments.

Step 2: FINN Technology Governance

- We also wanted to use it to build webapps
- Define it as an experiment in “FINN technology governance” model
  - Use on a few new non-critical services (Unleash, FINN Hjørner, Bedriftsprofiler)
  - Needed to reimplement tools (already implemented for java)
- Set-up internal npm repository

Step 3: Node Performance Rescue Squad

Step 4: Learn From the Best

Node Performance Workshop

- How to write performant Node.js applications
- How to debug Node.js in production?
  - Heap dumps
  - Flame charts
  - Remote debugging
- How to safely run Node.js applications in production

NODESOURCE™

Step 5: Educate the Organisation

Step 6: Standardize