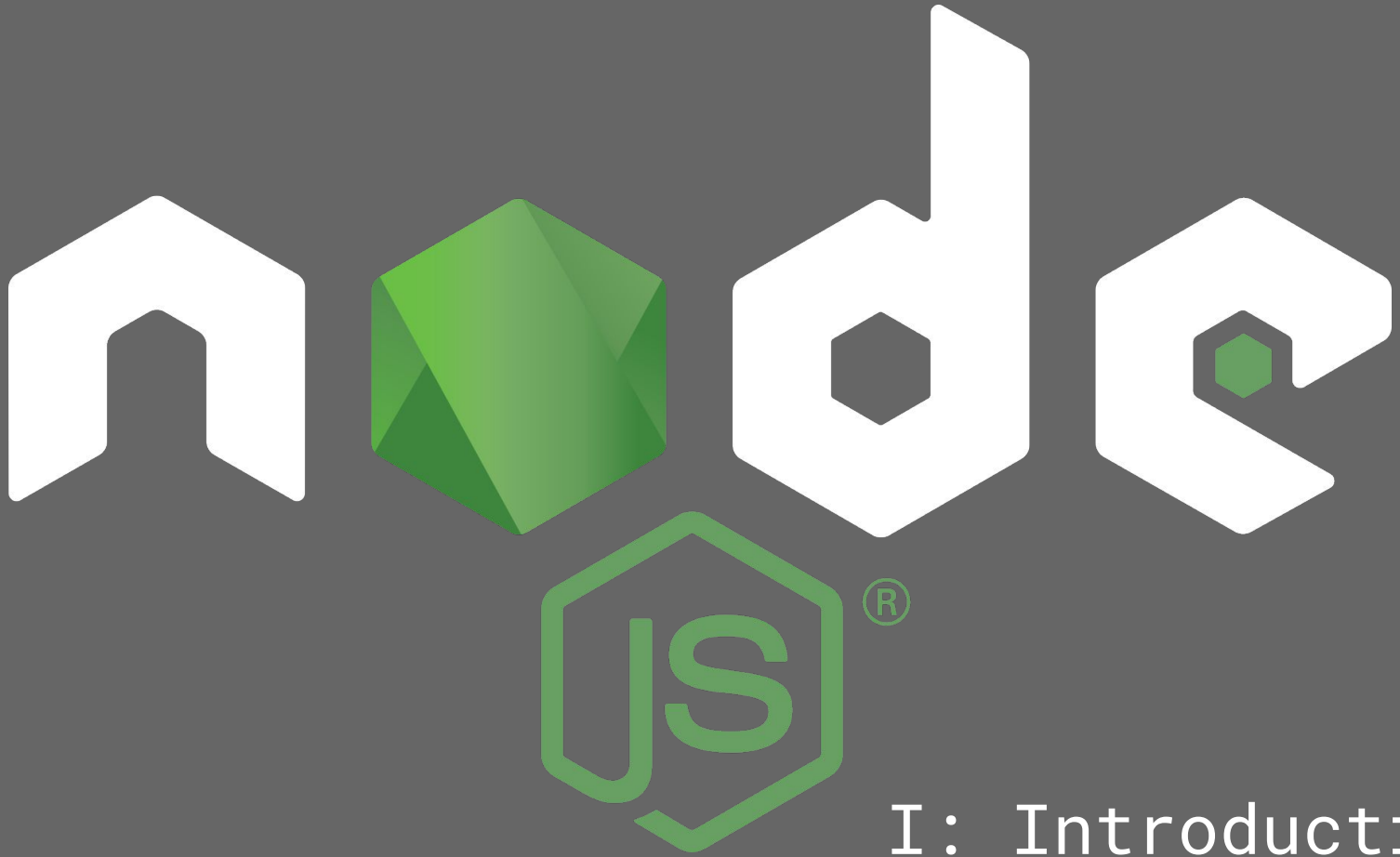


I: Introduction

What is Node.js?

Asynchronous, event-driven JavaScript runtime.

Asynchronous, event-driven JavaScript runtime.

Java is to the JRE is to the JVM

like

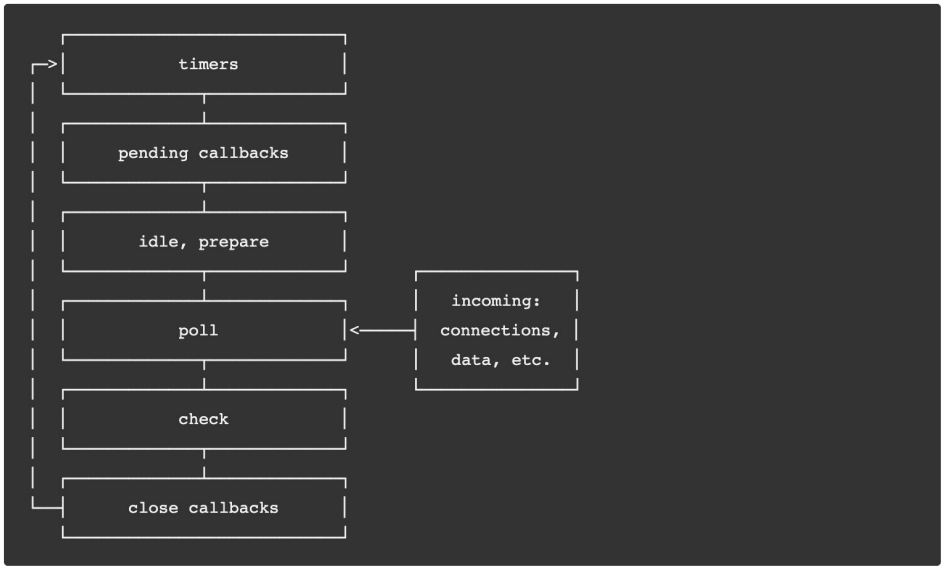
JavaScript is to Node.js is to V8.

Asynchronous, event-driven JavaScript runtime.

- Used to be interpreted
- Nowadays mostly just-in-time (JIT) compiled
- Node.js continuously compiles the program while running to further optimize
- Engine (e.g. V8) written in C++
  → JavaScript > V8(C++) > Machine Code
  → we can augment Node.js applications with C++ code if needed

Asynchronous, event-driven JavaScript runtime.

- Built-in event loop
- Similar in design to, and influenced by, systems like Ruby's Event Machine and Python's Twisted



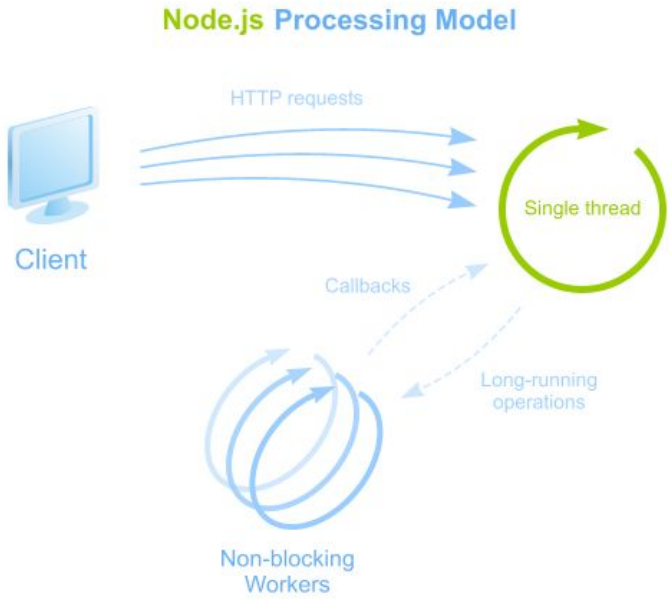
Asynchronous, event-driven JavaScript runtime.

- Non-blocking I/O by design

- `fs.readdir(path[, options], callback)`
- `fs.readdirSync(path[, options])`
- `fs.readFile(path[, options], callback)`
  - File Descriptors
- `fs.readFileSync(path[, options])`

I heard it’s single-process, single thread. WTF?!

- Thread-based networking is relatively inefficient and very difficult to use.
- Almost no function in Node.js directly performs I/O, so the process never blocks.



But what about my N processor cores?

Child processes can be spawned by using our `child_process.fork()` API, and are designed to be easy to communicate with. Built upon that same interface is the `cluster` module, which allows you to share sockets between processes to enable load balancing over your cores.

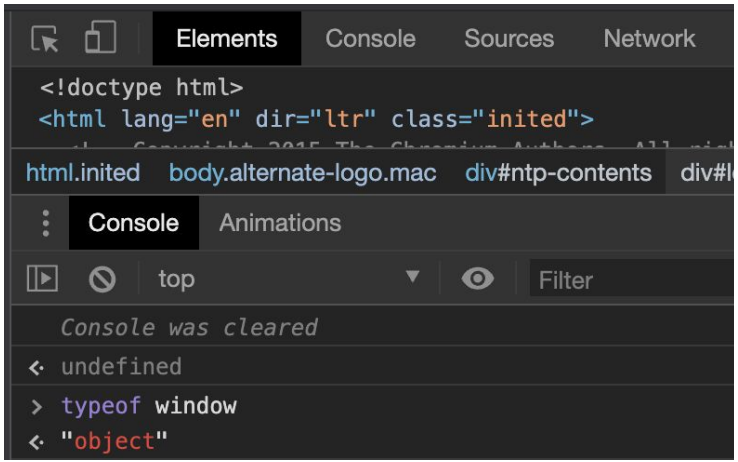
Hello...

What's the `diff` between Node.js and regular Browser stuff?

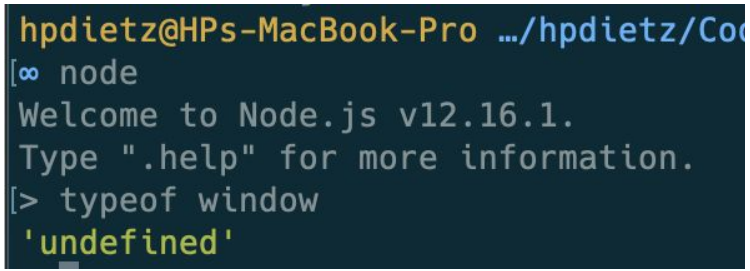
What's the diff between Node.js and regular Browser stuff?

Usually on Node.js…

- ...there is no `window`.



vs



What's the diff between Node.js and regular Browser stuff?

Usually on Node.js…

- ...there is no `window`.
- ...you use a different module system.

```
/components/SignupForm) - VIM3
import ReactGA from 'react-ga';
import { actions as signupFormActions } from '../../components/
SignupForm/actions';
import { actions as careerliftActions } from '../../components/
Careerlift/actions';
```

vs

```
rlift.jobs/src) - VIM3
1 const Koa = require('koa');
2 const serve = require('koa-static');
3 const compress = require('koa-compress');
4 const conditional = require('koa-conditional-get');
5 const etag = require('koa-etag');
```

What's the diff between Node.js and regular Browser stuff?

Usually on Node.js…

- ...there is no `window`.
- ...you use a different module system.
- ...you have access to all-the-things, e.g. file system, networking, etc.

```
fs.readFile(htmlTemplatePath, 'utf8');
```

```
11
12 server.listen(port, hostname, () => {
```

What's the diff between Node.js and regular Browser stuff?

Usually on Node.js…

- ...there is no `window`.
- ...you use a different module system.
- ...you have access to all-the-things, e.g. file system, networking, etc.
- ...you have native support for all the latest ECMA features.

```
const init = async ({
  htmlTemplatePath,
  stylesheetPath,
  selector
}) => {
```

```
async *linesGenerator() {
```

```
yield this.state.lines[currentLine];
```

```
await fs.readFile(htmlTemplatePath, 'utf8');
```

What’s the diff between Node.js and regular Browser stuff?

Usually on Node.js…

- ...there is no `window`.
- ...you use a different module system.
- ...you have access to all-the-things, e.g. file system, networking, etc.
- ...you have native support for all the latest ECMA features.
- ...you don’t care about bundling.

Why should I care?

- Using the same language on the client/frontend as and the server/backend.
- Sharing code between client and server.
- Rapid development.
- Built for state of the art distributed API architectures:

❖ *“HTTP is a first-class citizen in Node.js, designed with streaming and low latency in mind.”*

❖ *“Because nothing blocks, scalable systems are very reasonable to develop in Node.js.”*

❖ *“Node.js’ package ecosystem, npm, is the largest ecosystem of open source libraries in the world.”*