A bit about me

- Lived, taught and developed in Brighton for 20 years
- Recently relocated back to Northern Ireland
- Dad to two boys
- Love learning new things
- Passionate about JavaScript and the power it gives developers
- You can find me on Twitter (@dolearning) or on my website (https://kevincunningham.co.uk)

A bit about the course

Intro to Node	Binary data with Buffers
Node on the command line	Data processing with Streams
Core JS Concepts	Child processes
Packages and Dependencies	Testing with Node
Debugging	
Node module system	Express
Async control flow	RESTful Services
Error handling	SQL with Node

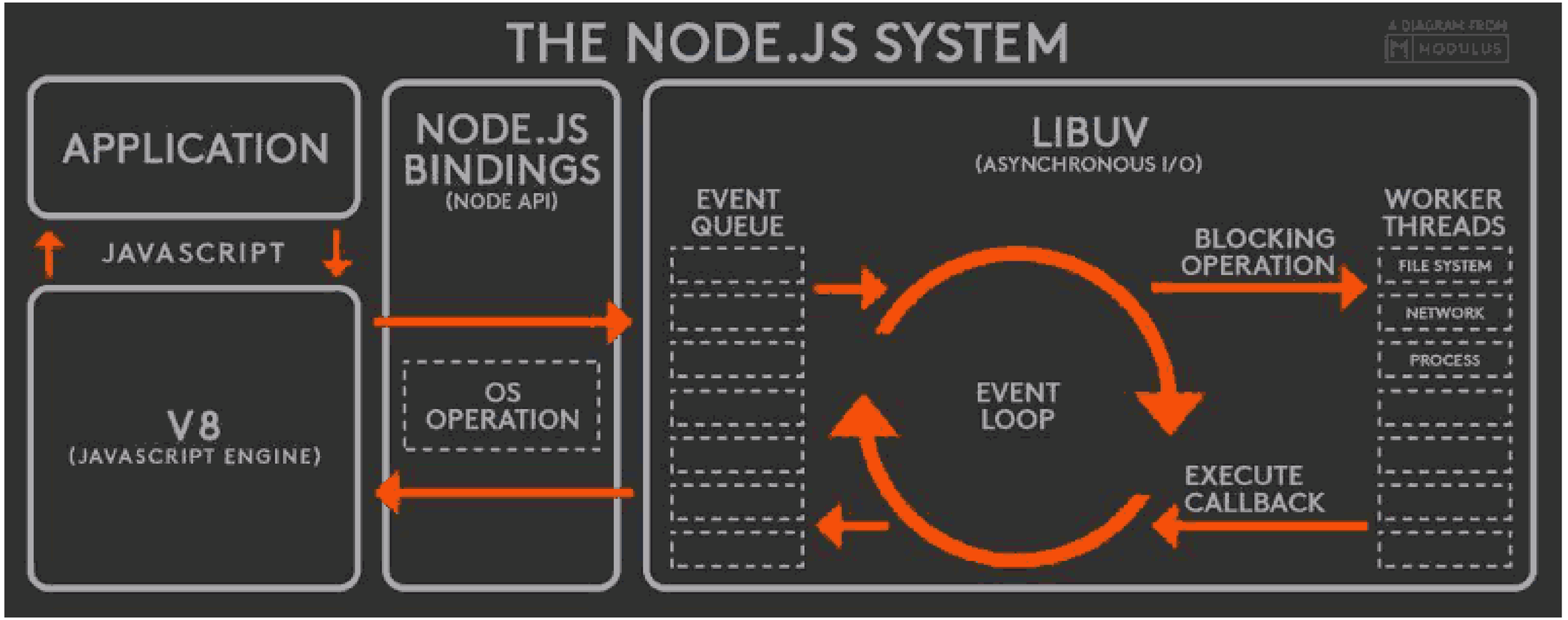
Timings

9.30 - 11	Session 1
11 - 11.15	Coffee
11.15 - 12.45	Session 2
12.45 - 1.45	Lunch
1.45 - 3.15	Session 3
3.15 - 3.30	Coffee
3.30 - 4.30	Session 4

A bit about you

- What your role is
- JavaScript and other programming experience
- What you’re hoping to get out of this course

What is Node.js?



Open source, free, cross-platform, JS on the server, event-driven, non-blocking, asynchronous, scalable

The Node.js Philosophy

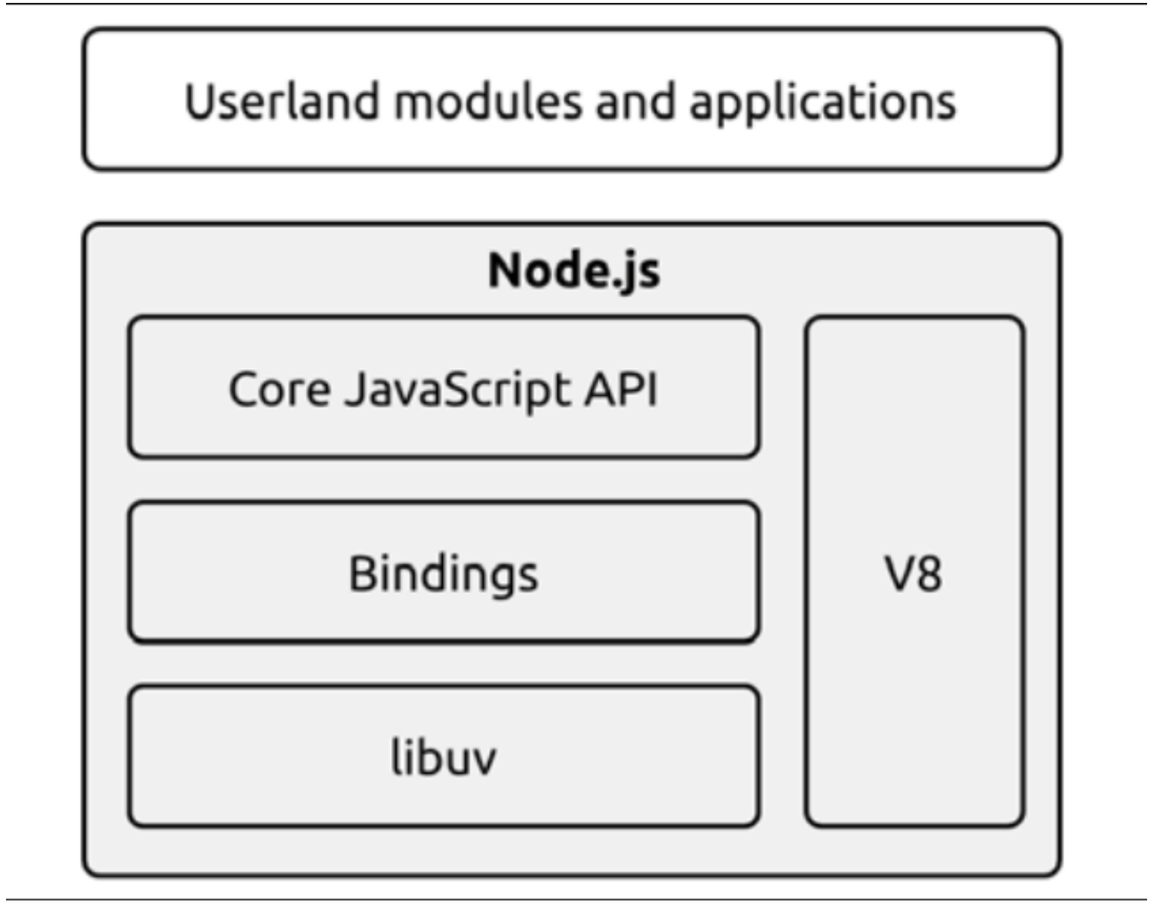
- Small core
- Small modules
- Small surface area
- Simplicity and pragmatism

How Node.js works

- I/O is slow
- Blocking I/O
- Non-blocking I/O
- Event demultiplexing
- The reactor pattern
- Libuv, the I/O engine of Node.js

The recipe for Node.js

- A set of bindings responsible for wrapping and exposing libuv
- V8, the JS engine developed by Google for Chrome. Acclaims for design, speed and memory managment.
- A core JS library that implements the high-level Node.js API



Node on the command line

By the end of this section, you should be able to:

- Explore all possible Node and V8 command line flags
- Use key utility mode command line flags
- Understand an essential selection of operational command line flags

- All command flags:
  - node --help
- V8 command flags:
  - node --v8-options
- Check syntax:
  - node --check app.js
- Dyanmic eval and print the result:
  - node --print
- Dynamic eval and don’t print the result:
  - node --eval
- Preload modules:
  - node --require
- Configure the stack trace:
  - node --stack-trace-limit

Node.js REPL

- `.editor`
- Double tab
- Ctrl-l

Editors

- Use what you want :)
- I use vim when I’m not teaching and VSCode when I am

First program

```
const http = require("http");

const hostname = "127.0.0.1";
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader("Content-Type", "text/plain");
  res.end("Hello, World!\n");
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Exercises (`Labs/01-introduction`)

1. In the labs folder, there is a file called `will-throw.js`. Run the file without any flags and then run with the appropriate flag to see the full call stack.

> In the first case, there should only be ten stack frames in the error output. In the second, there should be significantly more.

2. There are two other files in the labs folder, `bad-syntax.js` and `correct-syntax.js`. Use the appropriate flag to check the syntax of each file.

> There should be no output when checking `correct-syntax.js` and there should be a Syntax Error when checking the syntax of `bad-syntax.js`.

3. In the REPL, import the http library (`const http = require("http")`). Use the tab to explore some of the methods available. Also, check out the docs (https://nodejs.org/dist/latest-v16.x/docs/api/http.html)

4. Edit the server to send different responses, for example:

- Play with the status code
- Add some additional headers