# Node.js 101

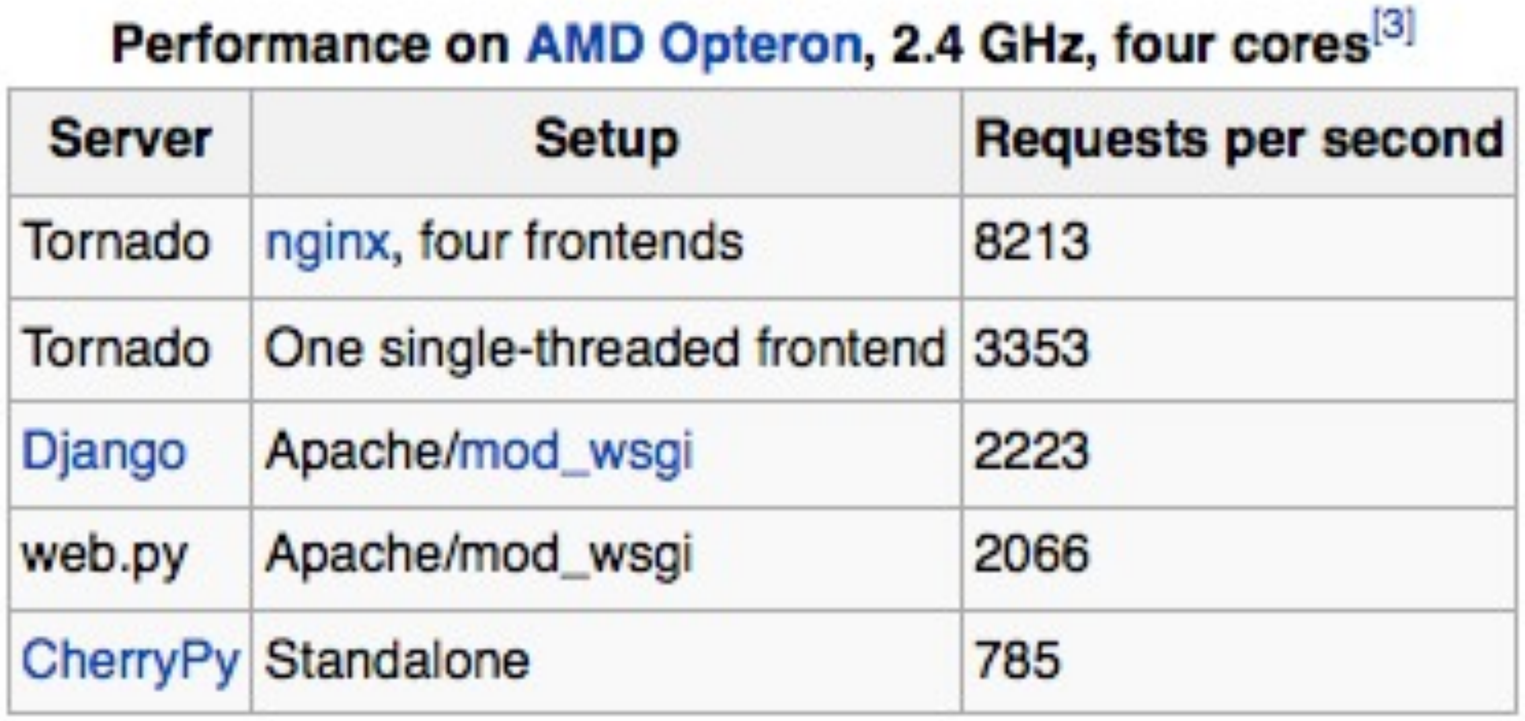
Hopefully, an intro to hacking on Node

## Misty origins of Node

- Initially released on 27 May 2009.
- Created by Ryan Dahl.
- His goal was to create web sites with push capabilities. He didn’t start with JS, but ended up there, due to lack of an existing I/O API. This allowed him to define the conventions of a non-blocking, event-driven I/O.
- (Plus, prolly, secretly, he knew JS is awesome).

## btdubs

- There are similar environments for other languages (non-blocking, event-driven).
- In particular, I’m thinking Tornado and Twisted for Python (cross-promotion for the other dev group).



**Performance on AMD Opteron, 2.4 GHz, four cores**[3]

| Server | Setup | Requests per second |
|---|---|---|
| Tornado | nginx, four frontends | 8213 |
| Tornado | One single-threaded frontend | 3353 |
| Django | Apache/mod_wsgi | 2223 |
| web.py | Apache/mod_wsgi | 2066 |
| CherryPy | Standalone | 785 |

## What Node is

- A platform for server-side and networking applications.
- These applications are written in JavaScript and executed by the Node.js runtime.
- Node applications are intended to maximize throughput and efficiency by using non-blocking I/O and asynchronous events.
- Callbacks in JS are executed during the course of the event loop.

## What Node is (cont’d)

- Node uses the Google V8 engine.
- V8 is fast, because it compiles JS to native machine prior to execution (not byte code or interpretation).
- It also optimizes the code at runtime based on heuristics of the code’s execution profile.

## Node architecture

- Four building blocks:
  - libuv to handle asynchronous events (C)
  - Google’s V8 run-time for JS
  - Core Node modules (http, assert, crypto), written in JS
  - Node bindings (C++) which are the connective tissue

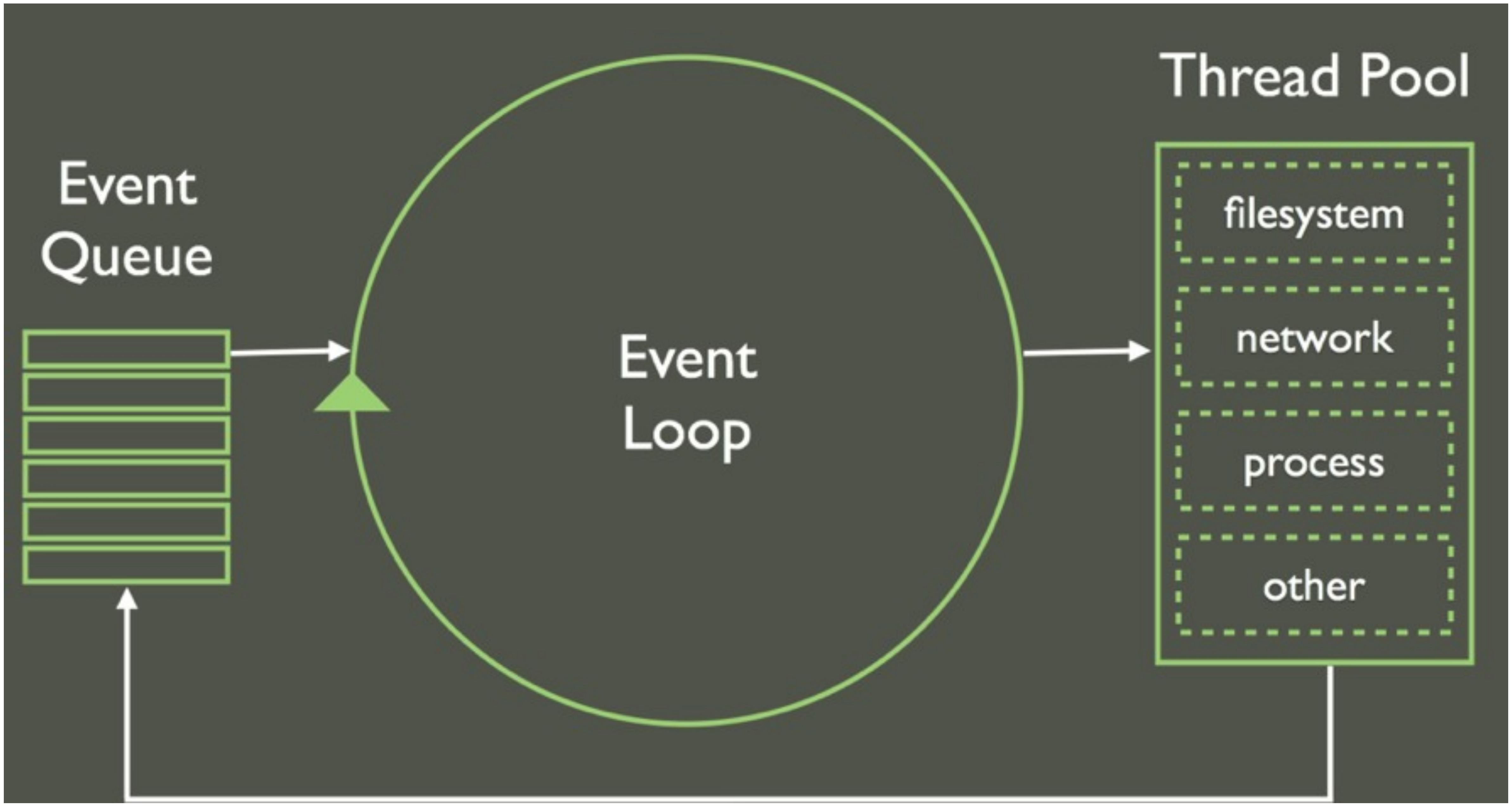
## Node architecture (tldr;)

- Basically, Node is a mix of C and JavaScript libraries that communicate.
- What does this mean for a dev? Probably not a lot, except...
    - You can do a whole bunch of things with JS and JS-like syntax that you couldn’t before
    - And it’s going to be fast

## Non-blocking?

- Node.js applications run single-threaded. This means a single line of JS at a time. It is not parallel.
- Thus, the need for a non-blocking I/O (via callbacks). Everything is asynchronous, which is different than client-side JS or server-side PHP.
- Multiple threads are used for file and network events and child processes can be created by devs (these do run in parallel).

## Event loop


Event Queue
Event Loop
Thread Pool
filesystem
network
process
other

# Asynchronous patterns

- Almost all of the interesting Node methods follow this pattern:
  - Last argument: callback
  - First callback argument: err
- fs.open(path, flags, [mode], callback)
- request('http://www.google.com', function (error, response, body) {});

## Insert...

... break here.

## Installation

- Installers are available for Windows (.msi) and Macs (.pkg), as well as pre-compiled binaries for Windows, Mac, Linux and SunOS.
- Node can also be installed via a package manager (including Homebrew and MacPorts).
- Finally, Node itself is open source and can be compiled on the target machine. For 'nix, this requires:
    - GCC 4.2+, GNU Make 3.81+
    - Python 2.6 or 2.7

## Installation (cont'd)

- For Windows:
  - Python 2.6 or 2.7
  - Visual Studio 2010 or 2012

http://nodejs.org/download/

https://github.com/joyent/node/wiki/Installing-Node.js-via-package-manager

https://github.com/joyent/node

## Installation (cont’d)

- Finally, you can use nvm to manage multiple Node installations (similar to rvm).

https://github.com/creationix/nvm

## Christmas presents!

- Node (node -v)
    - REPL
- npm (npm -v)

## REPL

- Read–eval–print loop; basically, a language shell.
- You can use the REPL from the command-line by typing node with no arguments.
- You can exit REPL by:
  - Typing .exit
  - Pressing Ctrl+C twice
  - Pressing Ctrl+D

## REPL (cont’d)

- The REPL provides access to any variables in the global scope.
- You can require local modules.
- _ contains the results of the last expression.
- A few more:
  - .break: Ditch a multi-line statement
  - .help: Show the list of special commands
  - .save: Save the current session to file

## JavaScript support

- Full ES5.
- Some ES6 (Harmony).
- No DOM.
- No window, but global is global.
    - The timers have been ported.
- process

## Modules

- Patterned after the CommonJS module system.
- Uses require(<module-name>).
  - Note! require is not the same as include.
- No DOM.
- Several core modules.
- Other modules can be added via npm.

## All too-brief intro to npm

- npm is Node’s package manager.
    - It does not stand for Node Package Manager!
- npm install <module-name>

## Whew...

Let’s code!

## Why we Node

No

Z
Z
Z