Introduction to Node.js

Introductions: About me

- John Paxton
- Email: pax@speedingplanet.com
- Live in New Jersey near NYC with my wife and hound dog
- 20+ years of web development experience

Introductions: About you

- Did I get your name right?
- Where are you in the world?
- What do you do at Northwestern Mutual?
- Why are you taking the class?
- What's a fun fact about you?

Tech check

- Node version?
- npm version?
- Git?
- Visual Studio Code or other editor

Class info

- John Paxton
- Repo: https://github.com/speedingplanet/nodejs-intro
- I'll make these slides available each day, becuase I usually make minor edits

Outline for today

- Setup (which we already finished)
- All about Node itself
- Node projects
- Testing with Node

Outline for the rest of class

- Events
- Input/output
- Filesystem
- Setting up a web server with Express
- Talking to databases with Object-relational mapping (ORM)
- Putting it all together as a REST server

What is Node.js and why is it different

- Stand-alone JavaScript engine
- Primarily used as a back-end application server
- Specialties
  - Asynchronous input/output
  - Event-driven architecture
- Open source project managed by the OpenJS Foundation

Versions

- The Node.js project actively maintains two versions
  - Long-term support (LTS)
  - Current development

Current Development

- Six month blocks
- Odd-numbered current versions go unsupported after 6 months
- Even-numbered current versions go to LTS
- Version 24 (24.6.0 as of August 19)
  - Goes to long-term-support in November

Long-term support

- Use these versions for production
- 30 months of active maintenance
- Sometimes backports of features from current releases

- Version 22 (22.18.0 as of August 19)
  - Maintenance mode in November
  - End of life mid-2027

Using different versions of Node

- We may need to host multiple versions of Node on a machine
- Two options: `nvm` and `n`
  - nvm (https://github.com/nvm-sh/nvm): OS-based
  - n (https://github.com/tj/n): Node-based

nvm

- Link: https://github.com/nvm-sh/nvm
- Primarily for Linux/Unix/MacOS-style systems
- Runs on typical Unix shells (bash, zsh, etc.)
- There is a Windows version nvm-windows
  - Not supported by the same group

n

- Link: https://github.com/tj/n
- Runs on top of Node
- But not system-agnostic: no Windows version
  - Unless you count Windows Subsystem for Linux

Key differences

- n keeps one primary install and caches other versions.
  - Switching with n is largely switching the node binary while keeping the environment
  - n's environment (global libraries, etc.) is always the same, regardless of version
- nvm keeps multiple install locations and swaps with symlinks/shortcuts
  - nvm allows you to use different versions (and environments) of node at the same time

The pillars of Node.js

- Google's V8 JavaScript engine
- The event loop

- Asynchronous coding

V8 JavaScript engine

- Node.js is a wrapped version of Google's V8 JavaScript engine
- The releases page has up-to-the-version info
- You can check your Node's version by running `node -p process.versions.v8` from the command line

The event loop

- Node is driven by events, rather than procedurally
  - Maps nicely onto functional programming, if you like that sort of thing
- Uses the event loop for scalability

Asynchronous coding

- Multi-threaded without the hassle
  - And in many cases, without the threads
- A simpler architecture for a very complex topic
- I/O tasks are asynchronous, but CPU tasks are not
  - Later, we will talk about a way to get around this!

Running Node.js applications

- Node has a REPL (Read Evaluate Print Loop)
  - Similar to a shell (bash, DOS, etc.)
  - Access by entering `node` at your command prompt
- Run one-liners from the command line
  - `node -e <code to evaluate>`
- `node <program.js>`
  - Many, many options, which we will touch on later

Exercise: experimenting with the interactive shell

- In this exercise, we will experiment with the REPL
- Enter the REPL by typing `node` at a command line

- Exit the REPL by typing `.exit` or Control-C twice
- Special commands begin with a dot
  - `.help` is useful
    - "dot-help"

Exercise continued

- Multiline statements are automatically detected by the REPL
  - Type Control-C once or .break to exist a multiline statement
- Overall readline-like support
  - Tab completion
  - History browsing by arrow keys
- Your instructor will guide you through several other features of the REPL