Node.js projects

Creating a project with NPM

- **N**ode **P**ackage **M**anager
- Create a project with the `npm` command
  - You can verify that NPM is installed by entering `npm --version`
- `npm init` will interactively set up a project for you
  - `npm init --yes` will create a project with all defaults accepted
- Packages vs projects
  - Functionally the same, it's legacy terminology

Exercise Creating a project with NPM

- Your instructor will walk you through project setup with `npm init`
- This will be a test project that we will later delete
  - But do not delete the project until the instructor says so!
  - We will need this project for at least one more exercise

Project structure

- `package.json` is the core configuration file
- Important parts
  - `dependencies` and `devDependencies`
  - `repository` and `description`
  - `scripts`
    - `npm start` is configured here
  - Can also contain other configuration

Project structure, continued

- node_modules
  - All dependencies wind up here
  - Organized and managed by npm (or other package manager, like yarn or pnpm)
  - Somtimes need to reset by deleting and re-installing
- .npmrc

    - Configure npm locally or globally with `npm config`
    - Versions for the project, the user, globally, and npm itself
    - Link: https://docs.npmjs.com/cli/configuring-npm/npmrc

Project structure, continued

- package-lock.json
    - Manages versions for different packages
    - Shouldn't be edited by hand

Dependencies

- Installing dependencies
    - `npm install <dependency>`
    - `npm i <dependency>`
    - `npm i <dependency>@<version>`
- Semantic versioning
    - major.minor.patch: more soon!
    - Incrementing major has to happen if backwards compatibility is broken

Dependencies, continued

- Tilde vs carat
    - `npm i lodash@~4.16` Install latest lodash 4.16; Accept patches to 4.16
    - `npm i lodash@^4` Install latest lodash 4; Accept updates <5.0.0
    - Slightly different rules for 0.x.x versions
- Finding available versions?
    - `npm view <package> versions` What versions are available?
    - `npm version <package>` What version of <package> is installed?

Types of dependency

- Runtime dependencies
    - Make the project work
    - Critical path
- Development dependencies
    - Support the project

    - Not needed to make the project work
    - Linters, servers, testing, etc.

Types of dependency, continued

- Peer dependencies
    - Dependency between a host package and a plugin or add-on
    - Expresses compatibility between packages
    - Even if there isn't a specific dependency
    - Warning if a peer dependency isn't provided

Exercise Installing dependencies

- In the same folder as the previous exercise, add some dependencies
- Start by installing just one dependency: lodash
    - Peek in the node_modules folder before and after the install: what changes?
- Install `http-server` as a development dependency
    - Try running `npx http-server` to see what happens
- Install bootstrap (a JS and CSS library)
    - What are you warned about?
    - How can you solve this issue?

Exercise Installing dependencies, continued

- Uninstall bootstrap
    - Are the other dependencies gone?
    - Clean up as necessary
- Install the latest available beta of Bootstrap 5
    - What shows up in your package.json?
    - Any new warnings?

Running packages

- You can run a package standalone with `npx <package name>`
    - Short for `npm exec` which it is mostly an alias to
- The package in question has to be able to run standalone
    - `lodash` or `bootstrap` do not have a standalone binary to run

    - `http-server` and `express` do
    - Check your package documentation

Semantic versioning

- Node uses semantic versioning for packages
- 3.2.1: major.minor.patch
- Patch updates
    - Do not break backwards compatibility
    - Usually just bugfixes and security updates
- Minor updates
    - Do not break backwards compatibility
    - Add features, roll up a succession of patches
- Continued next slide

Semantic versioning continued

- Major updates
    - BREAK backwards compatibility
    - You must increment the major version if you break backwards compatibility
    - This is not enforced by anything other than convention

Project tools

- You will want to use some tools with your JavaScript project
- Formatter: Prettier or similar
- ESLint for linting
- Testing: We will talk about testing in the next section

Exercise: Configuring our project

- We will install Prettier and ESLint
- Prettier doesn't require a lot of configuration
- ESLint does require some, but we'll use some established config files