The Javascript Package Ecosystem

A whirlwind tour for Rails developers

NPM - PACKAGE REPOSITORY & INSTALLER TOOL

NPM can mean 3 different things

- ✗ NPM is a for-profit company that was acquired by Github in 2020
- ✗ NPM is the javascript package repository, like RubyGems but for JS (node & frontend)
- ✗ NPM is a tool to manage packages in your project and on your computer
  - ✗ There’s a competing tool called yarn, too

A NOTE ON NODE

- NPM stands for ‘node package manager’.
- NodeJS is a javascript runtime, often used for servers
- An “asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications”
- It also works as a runtime, letting you run javascript command line utilities on your computer

Npm vs yarn for installing packages

- ✗ npm comes bundled with your node install
- ✗ yarn can be installed via npm, or with homebrew/apt:

```
npm install yarn -g
```

- ✗ yarn was originally made by facebook
  - ✗ It had some optimizations over npm that aren’t relevant any more
- ✗ Just pick one, and use it consistently within a project

YARN VS NPM - COMMANDS IN COMMON

```
[npm/yarn] init
```

- ✗ Initializes your project with a package.json and a lockfile

```
[npm/yarn] run [thing]
```

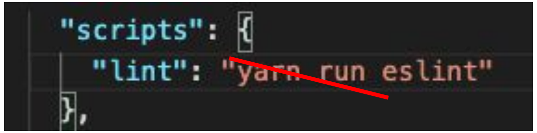
- ✗ Runs a package or a script

PACKAGE.JSON.WHAT??

- package.json is a very important file
  - It lists dependencies, like a Gemfile
  - It contains metadata about the project
  - It also can define scripts you can run
    - If you’re curious how to run the tests or the linter for some javascript, check the `scripts` section of package.json

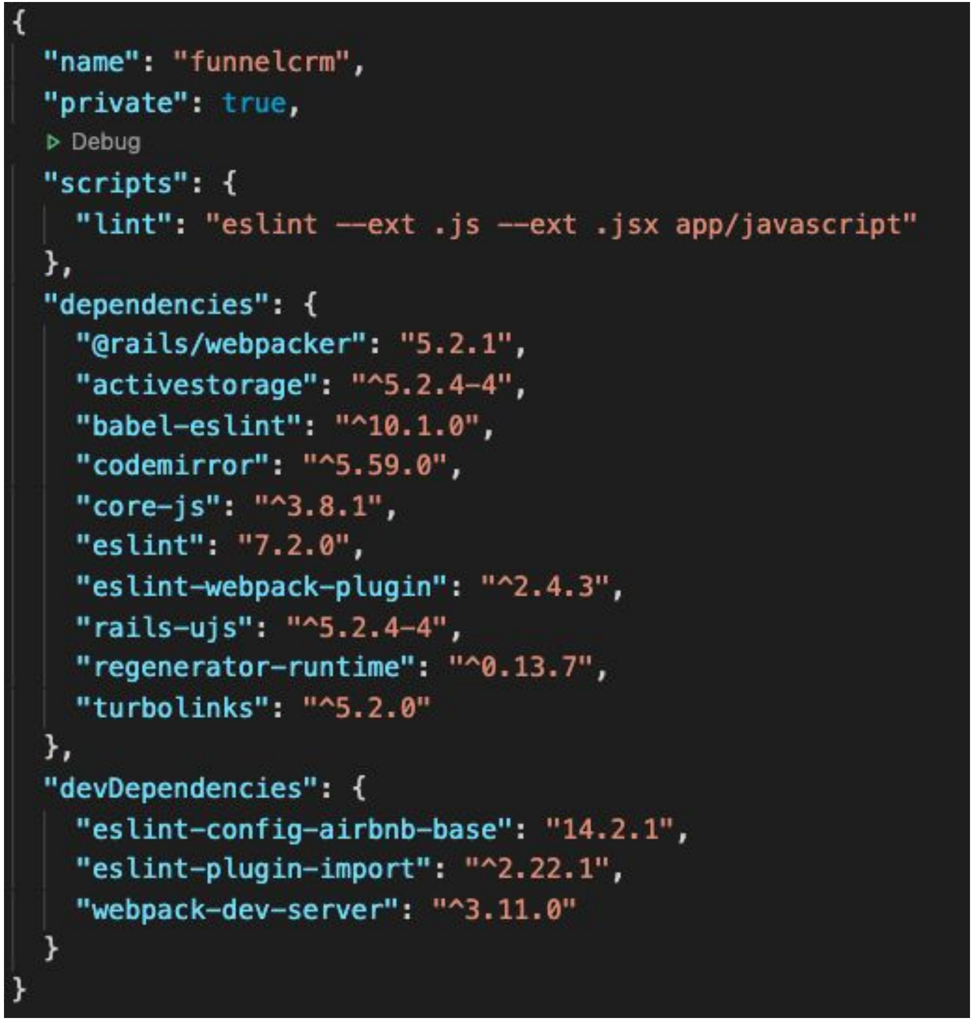
You’d run the lint script here with `yarn run lint` or `npm run lint`

Not a great example! Several things here are only needed in development, but I forgot to add the --dev flag so they’re in `dependencies` instead of `devDependencies`. But, doing so doesn’t really hurt much, especially for building apps, not packages to distribute



```
"scripts": {
  "lint": "~~yarn run~~ eslint"
},
```

Careful! You don’t need to put `yarn run` in the script definition



```
{
  "name": "funnelcrm",
  "private": true,
  "scripts": {
    "lint": "eslint --ext .js --ext .jsx app/javascript"
  },
  "dependencies": {
    "@rails/webpacker": "5.2.1",
    "activestorage": "^5.2.4-4",
    "babel-eslint": "^10.1.0",
    "codemirror": "^5.59.0",
    "core-js": "^3.8.1",
    "eslint": "7.2.0",
    "eslint-webpack-plugin": "^2.4.3",
    "rails-ujs": "^5.2.4-4",
    "regenerator-runtime": "^0.13.7",
    "turbolinks": "^5.2.0"
  },
  "devDependencies": {
    "eslint-config-airbnb-base": "14.2.1",
    "eslint-plugin-import": "^2.22.1",
    "webpack-dev-server": "^3.11.0"
  }
}
```

LOCKFILES

- Like Gemfile.lock
  - If using yarn, you’ll have a yarn.lock
  - If using npm, you’ll have a package-lock.json
- Should be checked into source control
  - Are a frequent source of annoying merge conflicts if multiple people added packages

Bundle install ::
npm install ::
yarn

ADDING A SINGLE PACKAGE (LIKE ESLINT)

`npm install eslint`	`yarn add eslint`
`npm install --save-dev eslint`	`yarn add eslint --dev`
`npm install -g eslint`	`yarn global add eslint`
`npm uninstall eslint`	`yarn remove eslint`
`npm update eslint`	`yarn upgrade eslint`

Peer dependencies

Peer dependencies: module A depends on module B, and it’s pretty likely that someone (Cassey) using A will already be using B, so module A says “I need module B to be installed if you’re going to use me”

Cassey installs module A, and gets a warning message that peer dependency B is missing and should be installed separately.
So Cassey installs B too, and then everything works*

* or you enter version hell

Other common tools

- For linting: **eslint**
- For transpilation: **babel**
  - Use newer JS language features that browsers don’t support yet, and transpile so it works in most browsers
  - Essential if you’re doing React with JSX; nice to have otherwise
- For code auto-formatting: **prettier**; **`eslint --fix`**
- For bundling: **webpack**
  - Handles things like running babel, autoprefixing CSS, putting all your JS in one file, imports, minifying, etc
  - Replaced grunt/gulp/a dozen other tools