Node Packages

What are Node Packages?

- Packages are one or more modules that have been packaged together and shared
- Most packages can be found on https://www.npmjs.com/, don't be afraid to search around!
- Your projects will technically be packages too, though you don't have to share them or let other people use them
- Information about your project, including the list of modules you use, are saved in a `package.json` file

`npm` Node Package Manager

Remember typing the **`npm`** *command when you installed RequireJS?*

- The NPM registry is a central place to get open-source modules, preventing you from reinventing the wheel
- Allows for installation from several types of location including the NPM registry, git, symbolic link, or a tarball archive
- Helps manage installed modules

`npm init`

- Creates a `package.json` for your project
- Follow the prompts to fill out information about your project. Don't worry, anything can be changed later and isn't too important.
- Saves all the information entered to the `package.json` file.

`package.json`

- Defines what packages your package depends on
- Defines the name and version of your package
- Defines what file gets executed if / when your package is required
- Holds basic licensing information (Can people use it, etc.)
- Gives author and repository information
- Optionally defines scripts and tasks (ie, `npm run [task]`)

`npm install:` local

- Installs packages locally to the `node_modules` folder
- Once installed, the package can be pulled into your source with `require` by name alone
- If you run `npm install` with `--save`, the dependency will be saved in your `package.json`. **Make sure you do this.**

```
# Download the jQuery module
npm install --save jquery
```

```
// Import the jQuery module
var $ = require("jquery");
```

`npm install:global`

- Installs the package globally, defaults to `.npm/` under your home directory
- Use the –g flag to denote a global install
- Should **only** be used for terminal tools, not for `require`

```
# install `http-server`, a tool that exposes
# folders and files over HTTP
npm install -g

# run `http-server` on your home directory on
# port, and opens it in your browser
http-server ~ -p 80 -o
```

npm uninstall

- Follows the same rules as `install`
- Uninstalls (removes) the package from `node_modules`
- `-g` uninstalls from the global packages
- `--save` will remove the dependency from `package.json`

```
npm uninstall --save jquery
npm uninstall -g http-server

http-server # command not found

var $ = require("jquery"); // Cannot find module 'jquery'
```

`npm ls`

- Shows a tree of the installed modules, meaning you can see your dependencies and their dependencies (and so on) at a glance
- Can be run with –g to show all globally installed modules
- Can be run with `--depth [number]` to limit how many dependencies-of-dependencies you see

```
npm ls

npm ls --depth 0
```

Challenge: Package-ize Star Printer

We're going to yet further enhance our star printer project by making it a package, and having it use a package.

1. Run `npm init` and fill out all of the prompts
2. Install the npm package `colors` (Don't forget to `--save`)
3. Make your stars print out with some kind of styling (It could just be one color, underlined, rainbow etc)

You can read how to use `colors` by looking at its documentation: https://www.npmjs.com/package/colors