Package Managers

Learning Objectives

- How to…
  - Install and remove modules
  - Update modules
  - All about package.json
  - Set init defaults
  - Local & global modules
  - Dependencies & dev dependencies
  - Listing modules
  - Semantic Versioning
  - NPM Scripts

Why?

Reusable/Modular Code

Package managers

Public & private repo

Re-Invent the wheel?

NPM Yarn

NPM

- Node Package Manager
  - It comes pre-installed with Node.js
  - A package manager is software that automates the process of installing, upgrading, configuring, and removing computer programs for a computer's operating system in a consistent manner. They are designed to eliminate the need for manual installs and updates.

- Allows us to install modules/package on your system
  - Modules are JavaScript libraries

Commands

- **npm -v** OR **npm --version**
  - Will tell us the version of npm that we are running on the computer

- **npm** OR **npm help**
  - Will both bring up a help page with useful commands
  - Mainly useful for building/publishing your own packages

- **npm init**
  - Initialize the package manager file (package.json)
  - We can use **npm init -y** to skip all the questions and create a default file

Package.json File

- Contains information about an the app you are building, including….
  - Lists dependencies (name & version) (& dev dependencies)
  - The npm scripts available on the package (and the ability to create more)
  - Created using **npm init**

- **npm set init-author-name "Sam Joyce"**
  - Can be used to alter the configuration of the default package.json file
  - **npm config delete init-author-name** to remove that

Dependencies

- A dependency is some third-party code that your application depends on.
- Just like a child depends on its parent, your application depends on other people's code.
- Our application needs this module/package in both the development stage (when we are working on it), as well as in production (when our users are using it).

Adding dependencies

npm install <package-name>

- **“install”** is a built in command that tells npm a package is being installed form the database of existing packages

- **“package-name”** will be matched with something from the npm database
  - https://www.npmjs.com

Package.json & node_modules relationship

- Modules are saved within the node_modules folder

- We therefore store our dependencies in our package.json and add the node_modules into a .gitignore which stops it being git tracked
    - If someone clones down our project they run one command to generate the node_modules
        - **npm install**

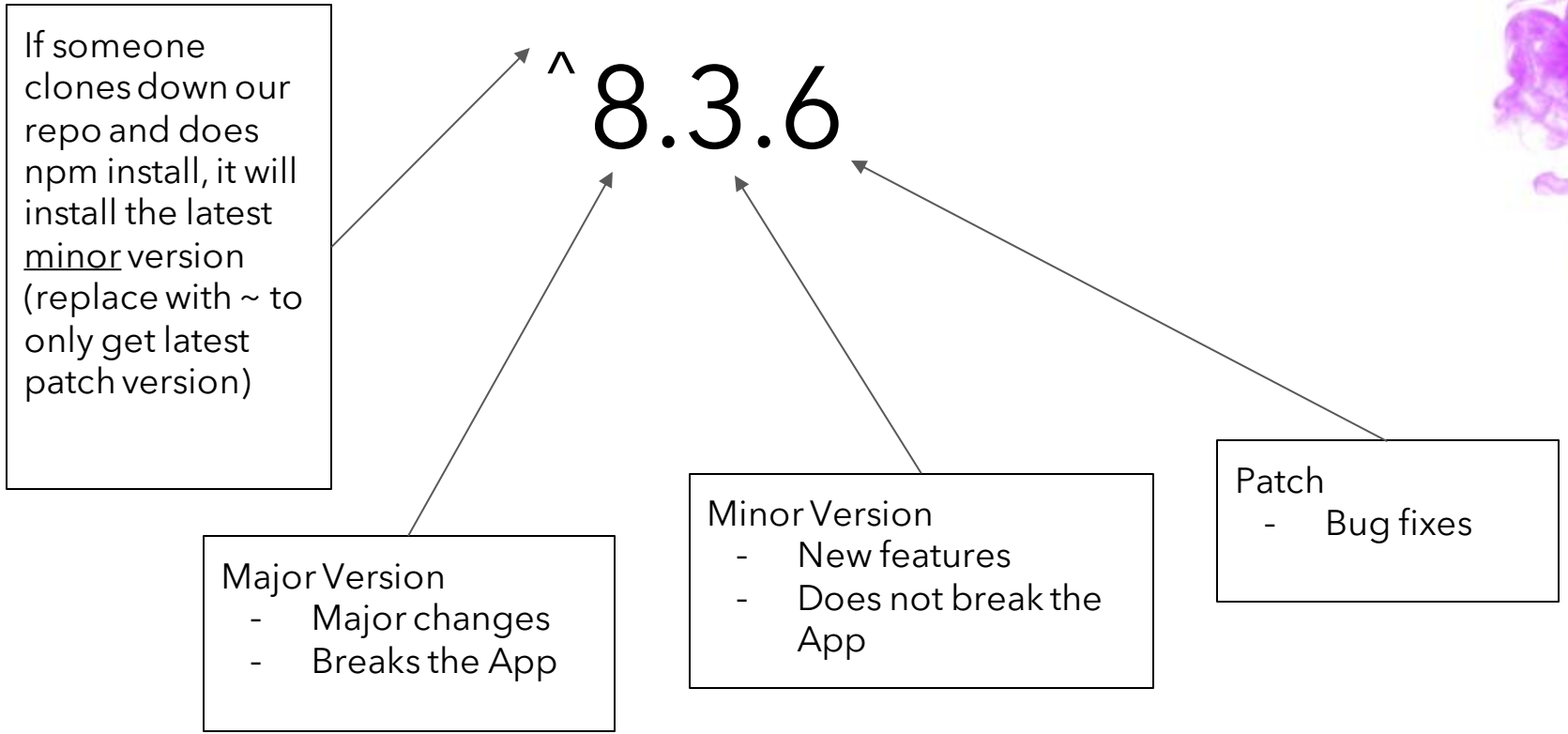
Commands Summary

- Adding a package
  - **npm install package-name**
  - Add -dev on the end for a dev dependency

- Removing a package
  - **npm uninstall package-name**
  - Add -dev on the end for a dev dependency

- Generate node_modules after cloning repo
  - **npm install**

Versioning


If someone clones down our repo and does npm install, it will install the latest minor version (replace with ~ to only get latest patch version)
^8.3.6
Major Version
- Major changes
- Breaks the App
Minor Version
- New features
- Does not break the App
Patch
- Bug fixes

Global Modules

- Instead of installing per project, we can install globally
  - To install a package globally
    - npm install -g package-name
  - To uninstall a package from global
    - npm uninstall -g package-name
  - To find the folder where global installs are
    - npm root -g

Scripts

- Allow us to define commands that run certain packages that we have installed
  - Install nodemon as a dependency
  - We can add a ‘watch’ script of ‘nodemon index.js’
  - When we enter **npm run watch** it will actually run our nodemon package on index.js

- We need to add run on everything except
  - start
  - test