NPM

Agenda

- Understand package management
- Discuss versioning issues
- A bit of mono repo challenges

Node Package Manager

- Introduced in 2011
- Automatically installed with Node
- "A set of publicly available reusable components, available through easy installation via an online repository, with version and dependency management"

- Full list of packages https://npmjs.org/

More

- Managed by npm, Inc.
- Company founded 2014 by npm's creator Isaac Z. Schlueter
- Runs the npm registry as free service
- Build supporting tools for the community
- Moto: "*when everyone else is adding force, we work to reduce friction*"

Not just for Node.js

- Using NPM you can install any package that adheres to NPM rules
- Not just node modules
- But also client side libraries
  - Angular
  - React
  - Vue

NPM repo

- npmjs.org is opened for every one to publish
- Is considered insecure
  - left-pad use case
- Can use your own local NPM repo
  - Artifactory
  - NPM Enterprise
  - Nexus
  - ProGet

npm -v

- Returns the version of currently installed npm
- Although installed together with Node.js, npm can be updated without updating node itself
- Node 8.10.0 is bundles with npm 5.6.0
- npm install –g npm
- Now, npm is version is 5.7.0

Global Installation

- npm install –g typescript
- Installs the package into the current logged-on user
  - Linux: /usr/local/lib/node_modules
  - Windows: %AppData%\npm\node_modules
- Global installation eventually creates conflicts between different projects
- Therefore, some consider that a bad practice

Local Installation

- npm install typescript
- Installs into the first parent directory that contains
  - node_modules
    - OR
  - package.json
- Therefore, directory structure is important
- Nested directory implies inheritance of packages

npm init

- Create package.json file inside the current directory
- Can append –y to skip all questions

```
{
  "name": “myapp",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "angular": "^1.6.9"
  }
}
```

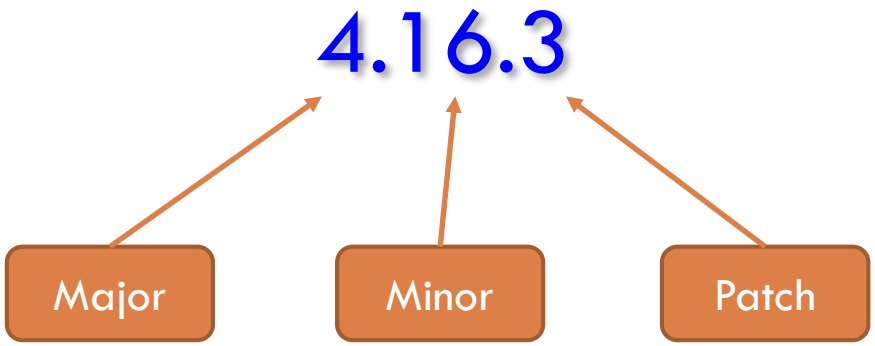
Updating package.json

- If package.json exists, npm automatically updates it with the installed package
  - Old npm did not do that

```
{
  "dependencies": {
    "express": "^4.16.3"
  }
}
```

SEMVER

- Semantic versioning



- Major - incompatible API changes
- Minor - New functionality in a backwards-compatible manner
- Patch - backwards-compatible bug fixes

Versioning Constraints

- ~version - Allows for greater patch number
- ^version – Greater minor number
- version – Exact match
- >version – Greater than

Lock down versions

- During development all team members should use the same version for all packages
- Can use exact match inside package.json
- However, it does not effect sub dependencies
- Starting NPM 5 a package-lock.json file is automatically created
  - Lists versions for all sub dependencies
  - You should commit it to source control

Common Dependency

- NPM will re-use common sub dependency as much as it can
  - Without breaking SEMVER
- The same package might reside multiple times inside node_modules
  - Each time with different version
- Might lead to multiple instances of the same package at runtime ☹

peerDependency

- npm does not install peer dependencies
- However it generates a warning
- The application owner is expected to install the missing dependency
- Use it when publishing a package that has dependency that is being used also by the application

devDependencies

- npm i typescript –save-dev
- Use it when installing packages that are not needed at runtime
  - Mostly build tools
- At production NPM does not install devDependencies
  - Production mode is considered such when NODE_ENV equals production

npm publish

- Publishes all files from current directory into the remote NPM repo
- Version inside package.json must not conflict with existing published packages
- Cannot overwrite existing package
- Consider cleaning directory before publishing it
- Need to login before publish
  - npm login

npm link

- Lets assume common and app packages
- Whenever common is changed we must
  - npm publish
  - npm install
- This flow is tedious and time consuming
- npm link allows linking both common and app directly without publishing to NPM repo
- Run npm link inside common and then npm link common inside app

Monorepo

- One repo with many projects
- Each project has its own package.json
- There are dependencies between projects
- Linking all projects is a nightmare
- NPM wont help ☹
- Yarn will ☺

Yarn workspace

- Install yarn
- Add package.json at the root directory that contains all projects → A.K.A workspace

```
{
 "private": true,
 "workspaces": [
  "common",
  "lib1",
  "lib2"
 ]
}
```

- Execute yarn
- All projects are now linked together

npm outdated

- The world is changing …
- Soon your project is out of date
- Must decide on the “upgrade approach”
- Upgrading all packages once a year creates a technical debt
- npm outdated is just the tip of the ice berg

Summary

- NPM is impressive
- The world largest open source repository
- You should use Yarn instead
  - Same internet repo
  - But mush faster
- Be careful of versioning hell