

NPM stands for Node Package Manager and it is the package manager for the Node JavaScript platform. It put modules in place so that node can find them, and manages dependency conflicts intelligently. Most commonly, it is used to publish, discover, install, and develop node programs.

Some Important npm commands every developer should know are:

- **NPM Install Command:** Installs a package in the *package.json* file in the local *node_modules* folder.

```
npm install
```

Example:

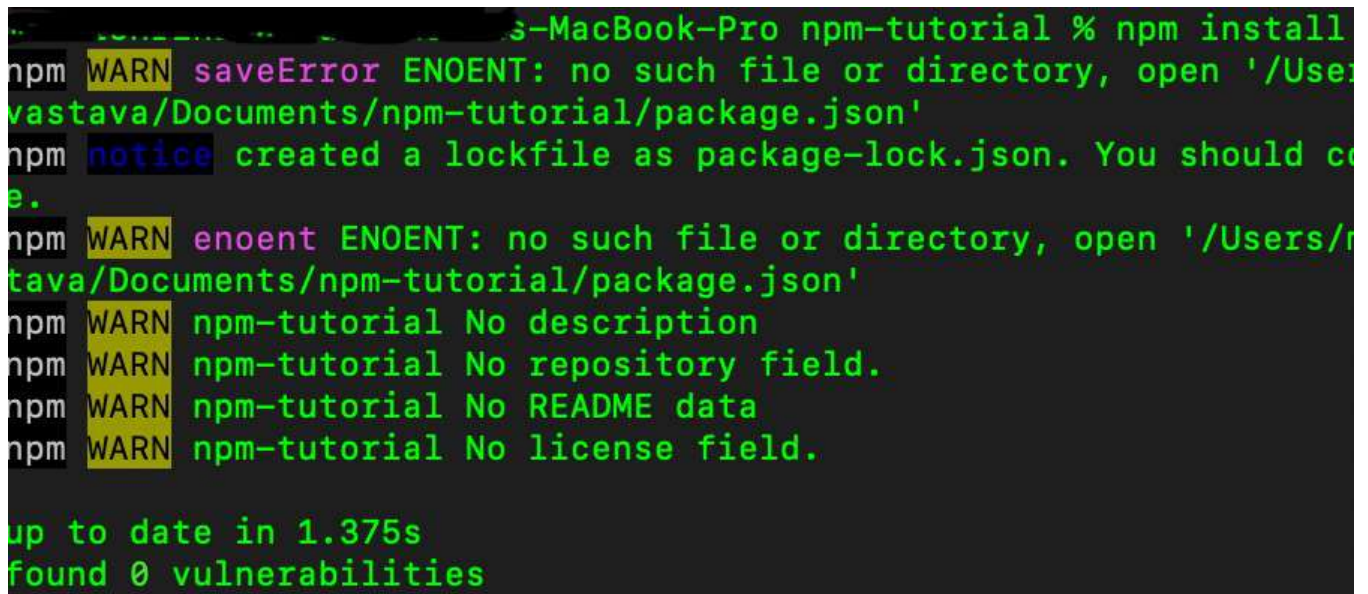
A terminal window on a MacBook Pro showing the output of the 'npm install' command. The prompt is 'npm-tutorial %'. The output includes several warnings: 'saveError ENOENT: no such file or directory, open \'/Users/.../Documents/npm-tutorial/package.json\'', 'notice created a lockfile as package-lock.json. You should commit this file.', 'enoent ENOENT: no such file or directory, open \'/Users/.../Documents/npm-tutorial/package.json\'', and four warnings about missing fields in package.json: 'No description', 'No repository field', 'No README data', and 'No license field'. The final output is 'up to date in 1.375s' and 'found 0 vulnerabilities'.

Image shows the use of "npm install" that install *package.json* and *package-lock.json*

- **NPM Uninstall Command:** Remove a package from the *package.json* file and removes the module from the local *node_modules* folder.

```
npm uninstall
```

Example:

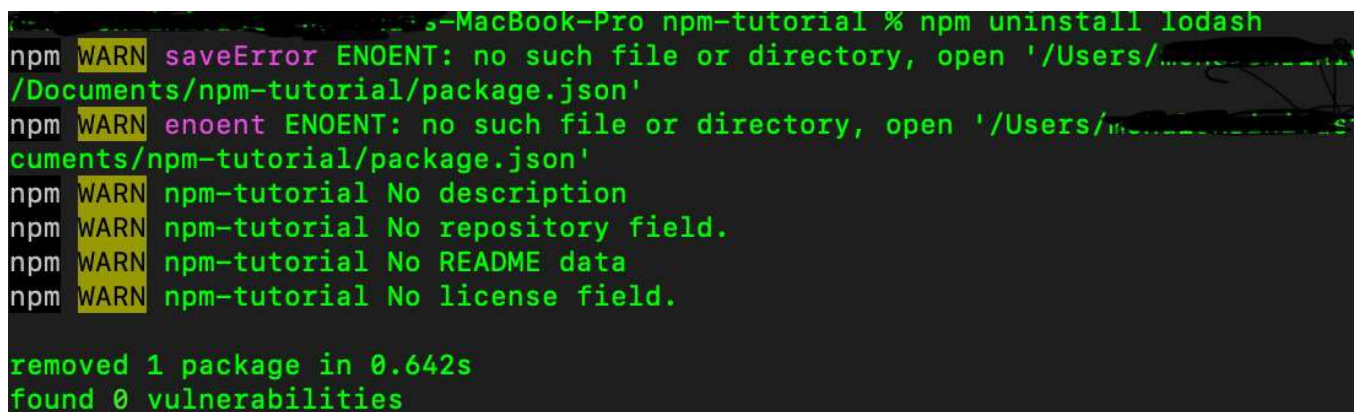
A terminal window on a MacBook Pro showing the output of the 'npm uninstall lodash' command. The prompt is 'npm-tutorial %'. The output includes several warnings: 'saveError ENOENT: no such file or directory, open \'/Users/.../Documents/npm-tutorial/package.json\'', 'enoent ENOENT: no such file or directory, open \'/Users/.../Documents/npm-tutorial/package.json\'', and four warnings about missing fields in package.json: 'No description', 'No repository field', 'No README data', and 'No license field'. The final output is 'removed 1 package in 0.642s' and 'found 0 vulnerabilities'.

Image shows a package 'lodash' which is an npm package being un-installed using *npm uninstall* command

- **NPM Update Command:** This command updates the specified package. If no package is specified then it updates all the packages in the specified location.

```
npm update
```

Example:

```
+ lodash@4.17.20
added 1 package from 2 contributors and audited 324 packages in 2.329s
found 0 vulnerabilities

MacBook-Pro npm-tutorial % npm update
npm WARN enoent ENOENT: no such file or directory, open '/Users/.../Documents/npm-tutorial/package.json'
npm WARN npm-tutorial No description
npm WARN npm-tutorial No repository field.
npm WARN npm-tutorial No README data
npm WARN npm-tutorial No license field.

+ lodash@4.17.21
updated 1 package and audited 324 packages in 1.156s
found 0 vulnerabilities
```

the original lodash version 4.17.20 -> updated to 4.17.21 using npm update command

- **NPM Global Update Command:** This command will apply the update action to each globally installed package.

```
npm update -g
```

Example:

```
MacBook-Pro npm-tutorial % sudo npm update -g
Password:
/usr/local/bin/npm -> /usr/local/lib/node_modules/npm/bin/npm-cli.js
/usr/local/bin/npx -> /usr/local/lib/node_modules/npm/bin/npx-cli.js
+ npm@6.14.13
updated 4 packages in 2.901s
```

npm update -g updates all of the packages if it's available.

- **NPM Deprecate Command:** This command will deprecate the npm registry for a package, providing a deprecation warning to all who attempt to install it.

```
npm deprecate
```

- **NPM Outdated Command:** Checks the registry if any (or specified) package is outdated. It prints a list of all packages which are outdated.

```
npm outdated
```

Example:

```
MacBook-Pro npm-tutorial % npm outdated
Package  Current  Wanted  Latest  Location
lodash   4.17.20  4.17.21  4.17.21  ls
MacBook-Pro npm-tutorial %
```

lodash package as indicated in the terminal is outdated that can be updated

- **NPM Doctor Command:** Checks our environment so that our npm installation has what it needs to manage our JavaScript packages.

```
npm doctor
```

Example:

```
MacBook-Pro npm-tutorial % npm doctor
npm notice PING https://registry.npmjs.org/
npm WARN verifyCachedFiles Content garbage-collected: 90 (13333068 bytes)
npm WARN verifyCachedFiles Cache issues have been fixed
Check          Value          Recommend
ation
npm ping       ok
npm -v        v6.14.13      Use npm v
7.10.0
node -v       v14.16.0      Use node
v14.16.1
npm config get registry https://registry.npmjs.org/
which git     /usr/bin/git
Perms check on cached files ok
Perms check on global node_modules ok
Perms check on local node_modules ok
Verify cache contents verified 4114 tarballs
```

- **NPM Initialize Command** Creates a package.json file in our directory. It basically asks some questions and finally creates a package.json file in the current project directory.

```
npm init
```



```

-MacBook-Pro npm-tutorial % npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults
.

See 'npm help init' for definitive documentation on these fields
and exactly what they do.

Use 'npm install <pkg>' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (npm-tutorial) geeksforgeeks
version: (1.0.0)
description: Tutorial for geeksforgeeks articles
entry point: (index.js) index.html
test command:
git repository:
keywords:
author: maestro1011
license: (ISC)
About to write to /Users/maestro1011/Documents/npm-tutorial/package
.json:
{
  "name": "geeksforgeeks",
  "version": "1.0.0",
  "description": "Tutorial for geeksforgeeks articles",
  "main": "index.html",
  "dependencies": {
    "live-server": "^1.2.1",
    "lodash": "^4.17.20"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "maestro1011",
  "license": "ISC"
}

Is this OK? (yes) yes

```

image shows the steps involved in npm init command.

- **NPM Start Command** Runs a command that is defined in the *start* property in the scripts. If not defined it will run the *node server.js* command.

```
npm start
```

- **NPM Build Command:** It is used to build a package.

```
npm build
```

Example:

```
MacBook-Pro npm-tutorial % npm build

New major version of npm available! 6.14.13 → 7.10.0
Changelog: https://github.com/npm/cli/releases/tag/v7.10.0
Run npm install -g npm to update!
```

Shows that there is a major update is available and can be updated using the command given after the changelog.

- **NPM List Command:** Lists all the packages as well as their dependencies installed.

```
npm ls
```

Example:

```
MacBook-Pro npm-tutorial % npm ls
geeksforgeeks@1.0.0 /Users/.../Documents/npm-tutorial
├── live-server@1.2.1
├── lodash@4.17.21
└── react-router-dom@5.2.0
```

npm ls lists all of the npm packages installed in the package.json file.

- **NPM Version Command:** Bumps a package version.

```
npm version
```

Example:

```

MacBook-Pro npm-tutorial % npm version
{
  geeksforgeeks: '1.0.0',
  npm: '7.10.0',
  node: '14.16.0',
  v8: '8.4.371.19-node.18',
  uv: '1.40.0',
  zlib: '1.2.11',
  brotli: '1.0.9',
  ares: '1.16.1',
  modules: '83',
  nghttp2: '1.41.0',
  napi: '7',
  llhttp: '2.1.3',
  openssl: '1.1.1j',
  cldr: '37.0',
  icu: '67.1',
  tz: '2020a',
  unicode: '13.0'
}

```

Lists out all packages version installed or used in the project.

- **NPM Search Command:** Searches the npm registry for packages matching the search terms.

npm search

```

MacBook-Pro npm-tutorial % npm search lodash

```

NAME	DESCRIPTION	AUTHOR	DATE
lodash	Lodash modular...	=mathias...	2021-
lodash-es	Lodash exported as...	=mathias...	2021-
@types/lodash	TypeScript...	=types	2021-
grunt	The JavaScript Task...	=cowboy =shama...	2020-
eslint-plugin-you-dont-ne	Check methods you...	=stevemao...	2021-
ed-lodash-underscore			
lodash.debounce	The lodash method...	=jldalton...	2016-
lodash.merge	The Lodash method...	=jldalton...	2019-
lodash.get	The lodash method...	=jldalton...	2016-
ext	JavaScript...	=tjholowaychuk	2019-
lodash.template	The Lodash method...	=jldalton...	2019-
lodash.isequal	The Lodash method...	=jldalton...	2017-
lodash.clonedeep	The lodash method...	=jldalton...	2016-
lodash.uniq	The lodash method...	=jldalton...	2016-
lodash.isplainobject	The lodash method...	=jldalton...	2016-
lodash.throttle	The lodash method...	=jldalton...	2016-
lodash.defaults	The lodash method...	=jldalton...	2016-
lodash.memoize	The lodash method...	=jldalton...	2016-
rambda	Lightweight and...	=self_refactor	2021-
lodash.sortby	The lodash method...	=jldalton...	2016-

shows the description of the package lodash and all commits and author who made the changes.

- **NPM Help Command:** Searches npm help documentation for a specified topic. It is used whenever the user needs help to get some reference.

npm help

Example:

```
-MacBook-Pro npm-tutorial % npm help
npm <command>

Usage:

npm install          install all the dependencies in your project
npm install <foo>    add the <foo> dependency to your project
npm test             run this project's tests
npm run <foo>        run the script named <foo>
npm <command> -h     quick help on <command>
npm -l              display usage info for all commands
npm help <term>     search for help on <term>
npm help npm        more involved overview

All commands:

access, adduser, audit, bin, bugs, cache, ci,
completion, config, dedupe, deprecate, diff, dist-tag,
docs, doctor, edit, exec, explain, explore, find-dupes,
fund, get, help, hook, init, install, install-ci-test,
install-test, link, ll, login, logout, ls, org, outdated,
owner, pack, ping, prefix, profile, prune, publish,
rebuild, repo, restart, root, run-script, search, set,
set-script, shrinkwrap, star, stars, start, stop, team,
test, token, uninstall, unpublish, unstar, update,
version, view, whoami
```

- **NPM Owner Command:** Manages ownership of published packages. It is used to manage package owners.

npm owner