Node Package Manager

Mohamed Mukthar Ahmed

1

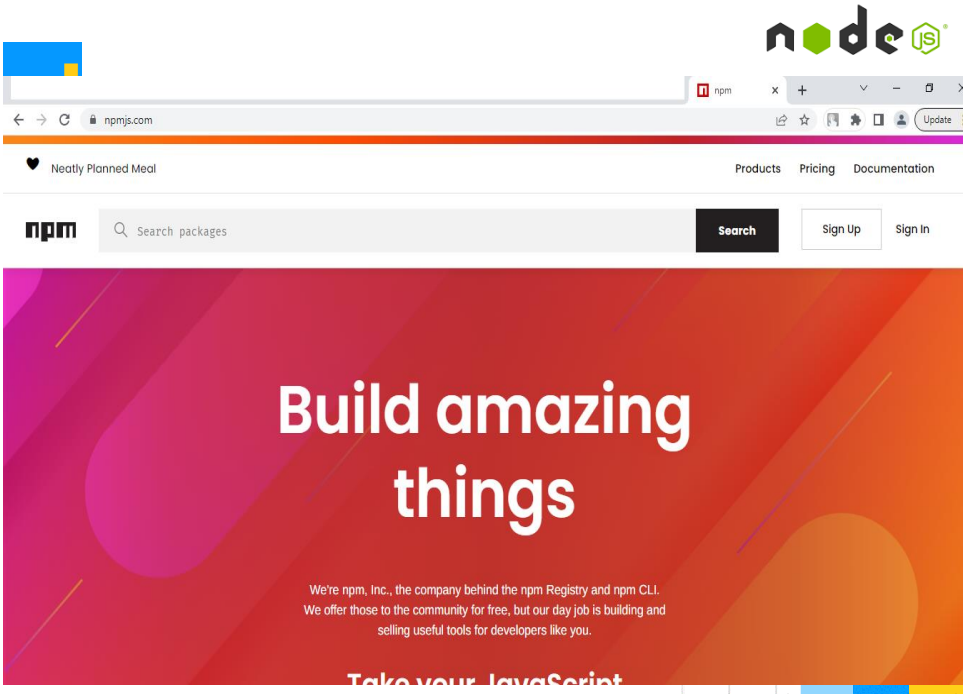
npm

Outline

- Intro to NPM
- Installing All Dependencies
- Installing Single Package
- Versioning
- Where NPM installs the package?
- Global Installation
- Updating Packages
- The package.json Guide
- The package.json Content
- Find Installed Packages
- Uninstall npm Packages
- Node Package Runner

Mohamed Mukthar Ahmed

2



Mohamed Mukthar Ahmed

3

Intro to npm

- **Node Package Manager** (npm) is the **standard** package manager for Node.js.
- In **January 2017** over **350000** packages were reportedly being listed in the npm registry, making it the biggest single language code repository on Earth.
- It started as a way to download and manage dependencies of Node.js packages, but it has since become a tool used also in frontend JavaScript.
- **NOTE**: **Yarn** and **pnpm** are alternatives to **npm** command line interface.

Mohamed Mukthar Ahmed

4

Installing All Dependencies

- **Node Package Manager** (npm) is the **standard** package manager for Node.js.
- If a project has a **package.json** file, by running

```
C:\thePrj>npm  install
```

- it will install everything the project needs, in the **node_modules** folder, creating it if it's not existing already.

5

Installing Single Package

- You can also install a **specific package** by running

```
C:\thePrj>npm  install  <package_name>
```

- Furthermore, since **npm 5**, this command adds <package-name> to the **package.json** file dependencies.
- Often, you'll see more **flags** added to this command:

Common npm Flags	
Flag	Description
-g	Install globally
-D	Install and add the entry to package.json file devDependencies
-O	Install and add the entry to package.json file optionalDependencies

6

Installing Single Package

- The difference between **devDependencies** and **dependencies** is that the former contains development tools, like a testing library, while the latter is bundled with the app in production.
- As for the **optionalDependencies** the difference is that build failure of the dependency will not cause installation to fail. But it is your program's responsibility to handle the lack of the dependency.

Mohamed Mukthar Ahmed

7

Versioning

- **npm** also manages **versioning**, so you can specify any specific version of a package
- Specifying an **explicit version** of a library helps to keep everyone on the same exact version of a package, so that the whole team runs the same version until the **package.json** file is updated.
- In such cases, versioning helps a lot, and npm follows the **semantic versioning** (**semver**) standard.
- You can install a specific version of a package, by running

```
C:\thePrj>npm  install  <package_name>@<version>
```

Mohamed Mukthar Ahmed

8

Where does npm install the packages?

- When you install a package using **npm** you can perform TWO types of installation:
  - a **local** install
  - a **global** install
- By default, when you type an npm install command, like:

```
C:\thePrj>npm  install  lodash
```

- the package is installed in the current file tree, under the **node_modules** subfolder.
- As this happens, **npm** also adds the **lodash** entry in the dependencies property of the **package.json** file present in the current folder.

Mohamed Mukthar Ahmed

9

Where does npm install the packages?

```
C:\thePrj>npm  install  lodash
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN theprj@1.0.0 No description
npm WARN theprj@1.0.0 No repository field.

+ lodash@4.17.21
added 1 package from 2 contributors and audited 1 package in 1.67s
found 0 vulnerabilities
```

```
C:\thePrj>tree
Folder PATH listing for volume OS
Volume serial number is 9EAD-1162
C:.
└───node_modules
    └───lodash
        └───fp
```

Mohamed Mukthar Ahmed

10

Global Installation

- A global installation is performed using the **-g** flag
- When this happens, **npm** won't install the package under the local folder, but instead, it will use a global location.

Where, exactly?

- The **npm root -g** command will tell you where that exact location is on your machine.
- On **macOS** or **Linux** this location could be **`/usr/local/lib/node_modules`.**
- On **Windows** it could be C:\Users\YOU\AppData\Roaming\npm\node_modules
- If you use **nvm** to manage Node.js versions, however, that location would differ.

Mohamed Mukthar Ahmed

11

Updating Packages

- Updating is also made easy, by running

```
npm update [-g] [<pkg>...]

aliases: up, upgrade
```

- **npm** will check all packages for a newer version that satisfies your versioning constraints.

Running Tasks

- The **package.json** file supports a format for specifying command line tasks that can be run

Mohamed Mukthar Ahmed

12

The package.json Guide

- If you work with JavaScript project, Node.js or a frontend project, you surely would have seen the **package.json** file.
  - What's that for?
  - What should you know about it? and
  - What are some of the cool things you can do with it?
- The **package.json** file is kind of a **manifest** for your project. It can do a lot of things, completely unrelated. It's a central repository of configuration for tools, for example.
- It's also where **npm** and yarn store the names and versions for all the installed packages.

Mohamed Mukthar Ahmed

13

The package.json Content

- There are lots of things going on here:
- **version** indicates the current version
- **name** sets the application/package name
- **description** is a brief description of the app/package
- **main** sets the entry point for the application
- **private** if set to true prevents the app/package to be accidentally published on **npm**
- **scripts** defines a set of node scripts you can run
- **dependencies** sets a list of **npm** packages installed as dependencies

Mohamed Mukthar Ahmed

14

The package.json Content

- There are lots of things going on here:
- **devDependencies** sets a list of **npm** packages installed as development dependencies
- **engines** sets which versions of Node.js this package/app works on
- **browserslist** is used to tell which browsers (and their versions) you want to support
- All those properties are used by either npm or other tools that we can use.

Mohamed Mukthar Ahmed

15

Find Installed npm Packages

- To see the version of all installed **npm** packages, including their dependencies

```
npm list
```

```
> npm list
/Users/joe/dev/node/cowsay
└─┬ cowsay@1.3.1
  ├── get-stdin@5.0.1
  ├─┬ optimist@0.6.1
  │ ├── minimist@0.0.10
  │ └── wordwrap@0.0.3
  ├─┬ string-width@2.1.1
  │ ├── is-fullwidth-code-point@2.0.0
  │ └─┬ strip-ansi@4.0.0
  │   └── ansi-regex@3.0.0
  └── strip-eof@1.0.0
```

Mohamed Mukthar Ahmed

16

Find Installed npm Packages

- You can also just open the **package-lock.json** file, but this involves some visual scanning.
- **`npm list -g`** is the same, but for globally installed packages.
- To get only your top-level packages (basically, the ones you told **npm** to install and you listed in the package.json), run **`npm list --depth=0`**

```
> npm list --depth=0
/Users/joe/dev/node/cowsay
└── cowsay@1.3.1
```

```
> npm list cowsay
/Users/joe/dev/node/cowsay
└── cowsay@1.3.1
```

```
> npm view cowsay version

1.3.1
```

Mohamed Mukthar Ahmed

17

Uninstalling npm Packages

- To uninstall a package, you have previously installed locally (using npm install <package-name>), run

```
C:\thePrj>npm  uninstall  <package_name>
```

- If the package is installed globally, you need to add the –g option

```
C:\thePrj>npm  uninstall  <package_name> -g
```

Mohamed Mukthar Ahmed

18

The npx Node.js Package Runner

- **npx** is a very powerful command that's been available in **npm** starting version **5.2**, released in July 2017.
- **NOTE**: If you don't want to install **npm**, you can install **npx** as a standalone package
- **npx** lets you run code built with Node.js and published through the **npm** registry.

Mohamed Mukthar Ahmed

19

Installation-less Command Execution

- There is great feature of **npx**, which is allowing to run commands without first installing them.
- This is pretty useful, mostly because:
  - you don't need to install anything
  - you can run different versions of the same command, using the syntax **@version**
- A typical demonstration of using **npx** is through the **cowsay** command. **cowsay** will print a cow saying what you wrote in the command.

Mohamed Mukthar Ahmed

20

Installation-less Command Execution

- **cowsay** will print a cow saying what you wrote in the command.

```
C:\thePrj>
C:\thePrj>npx cowsay "Hello"
npx: installed 41 in 9.671s
 _______
< Hello >
 -------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||

C:\thePrj>tree
Folder PATH listing for volume OS
Volume serial number is 00000232 9EAD:1162
C:.
└───node_modules
    └───lodash
        └───fp
```

Mohamed Mukthar Ahmed

21

Mohamed Mukthar Ahmed

22