FREE eBook

LEARNING

npm

Free unaffiliated eBook created from
Stack Overflow contributors.

#npm

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: npm

It is an unofficial and free npm ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official npm.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with npm

Remarks

npm is the default package manager for Node.js. It is written entirely in JavaScript, and allows to install and distribute mostly JavaScript modules on the registry. Packages are treated in the CommonJS format, and are recognizable by the presence of a `package.json` file.

Versions

Version	Release Date
v4.0.0	2016-10-21
v3.10.0	2016-06-17
v3.9.0	2016-05-06
v2.15.0	2016-03-11
v3.8.0	2016-02-26
v3.7.0	2016-01-29
v3.6.0	2016-01-21
v3.5.0	2015-11-20
v3.4.0	2015-11-06
v3.3.0	2015-08-14
v2.14.0	2015-08-14
v3.2.0	2015-07-25
v3.1.0	2015-07-03
v2.13.0	2015-07-03
v3.0.0	2015-06-26
v2.12.0	2015-06-19
v2.11.0	2015-05-22
v2.10.0	2015-05-08

Version	Release Date
v2.9.0	2015-04-24
v2.8.0	2015-04-10
v2.7.0	2015-02-27
v2.6.0	2015-02-13
v2.5.0	2015-01-30
v2.4.0	2015-01-23
v2.3.0	2015-01-16
v2.2.0	2015-01-09
v2.1.0	2014-09-26
v2.0.0	2014-09-13
v1.4.0	2014-02-13

Examples

Installation or Setup

Install

`npm` is bundled with Node.js, so if you install Node.js you'll automatically have `npm` installed too. You can choose between a **Current** and a **LTS** version

Windows

For Microsoft Windows you can download a **MSI installer** from https://nodejs.org/en/download/.

OS X

For Apple OS X you can download a **PKG installer** from the same location https://nodejs.org/en/download/.

Linux

For Linux you can use your package manager to install Node.js and npm.

You can also compile Node.js from source and you'll still get `npm`. There is also a script you can run which will install `npm`:

```
curl -L https://www.npmjs.com/install.sh | sh
```

Upgrade npm to the latest version

The recommended method of updating your `npm` installation is to simply have `npm` install itself:

```
npm install -g npm@latest
```

You can alternatively upgrade to the current LTS version rather than the latest version:

```
npm install -g npm@lts
```

You can also install any version of Node (and npm) with `nvm`. When installing globally with `npm` with an `nvm` installation, you do not need to use `sudo` (or Run as Administrator on Windows).

Install packages

Notice that packages can be installedThis command installs the newest available version of the named packages:

both locally or globally.

Local installation means that **npm** installs your package in the current working directory. Node modules go in `./node_modules`, executables go in `./node_modules/.bin/`. Usually you'll want to install local modules for usage inside your program, as a dependency, and they will work only on where they're installed.

```
npm install <package names>
```

Shorthand:

```
npm i <package names>
```

`npm` can interact with a `package.json` file in the current directory in various useful ways, through the objects `dependencies` and `devDependencies` stored in `package.json` (installing multiple modules):

The `npm install` command with no parameters

```
npm install
```

installs all packages named as object keys in the `dependencies` and `devDependencies` objects in `package.json`, using semantic versioning restrictions as indicated by the object values.

When developing new software:

Use option `-S` to append the `<package names>` and versions of npm modules you are installing that should always be included with your module. Appends to the list of `dependencies` tracked in the

`package.json` file, after installing.

```
npm i <package names> -S
```

Use option `-D` to append the `<package names>` and versions of npm modules you are installing that are needed by other developers to further develop or test your module. Appends to the list of `devDependencies` tracked in the `package.json` file, after installing.

```
npm i <package names> -D
```

Where `lodash` and `mocha` are package names.

Installing Global Packages

Install a global package

Globally installed packages drops modules in `{prefix}/lib/node_modules`, and puts executable files in `{prefix}/bin`, where `{prefix}` is usually something like `/usr/local`. Installing a global module means that its binaries end up in your `PATH` environment variable. Usually you'll want to install a global module if it's a command line tool, or something that you want to use in your shell.

```
npm install --global package-name
```

Remove a global package

```
npm uninstall --global package-name
```

Note: the `--global` argument can be simplified to `-g`. So, for instance, the first command could have been `npm install -g package-name`, with the exact same outcome.

Note: in *nix systems, installing global packages may require super-user permissions. Failing to do so will fail with: `EACCES`. In that case, run:

```
sudo npm install --global package-name
```

Updating packages

In every applications life-cycle comes a day where it's components needs to be updated. Everyone knows the pain of updating every single dependency one-by-one. Well here you just need to issue the command:

```
npm update (-g)
```

If the "-g" is there then npm will update the global packages.

Using npm to manage dependencies

So you want to deploy your app to multiple sites? and your project has too many dependencies to install them one-by-one? Npm has a solution just issue the following command:

```
npm init
```

In the project's root folder then follow the instructions on screen (type in the desired value then press enter) and then if you want to save a dependency then add:

```
--save
```

after your

```
npm install
```

commands for example:

```
npm install mypackagename --save
```

And then that dependency will be saved then you don't have to move the "node_modules" folder. In order to install all saved dependency issue:

```
npm install
```

and all saved dependencies will be installed.

Read Getting started with npm online: https://riptutorial.com/npm/topic/2061/getting-started-with-npm

Chapter 2: npm scripts

Syntax

- The "scripts" property in `package.json` allows you to run npm packages locally.
- The `"karma": "karma"` script references the `karma` shell script is the `node_modules/.bin` directory. This reference needs to be grab and an alias needs to be applied to it in order to be used in other npm scripts, such as `"test": "karma start"`.

Remarks

Pre-recognized scripts

- `prepublish`: Run before the package is published
- `publish`, `postpublish`: Run after the package is published
- `preinstall`: Run before the package is installed
- `install`, `postinstall`: Run after the package is installed
- `preversion`, `version`: Run before bump the package version
- `postversion`: Run after bump the package version
- `pretest`, `test`, `posttest`: Run by the `npm test` command
- `prestop`, `stop`, `poststop`: Run by the `npm stop` command
- `prestart`, `start`, `poststart`: Run by the `npm start` command
- `prerestart`, `restart`, `postrestart`: Run by the `npm restart` command. Note: `npm restart` will run the stop and start scripts if no `restart` script is provided.

It can be deduced that the `"scripts"` property in `package.json` is a very powerful tool. It can be used as a build tool, similar to the likes of Grunt and Gulp, but with over 250,000 packages available. NPM scripts runs npm packages installed locally to your project from the `node_modules/.bin` directory.

Examples

Running karma locally

`package.json` snippet

```
{
    "scripts":
        "test": "karma start",
        "karma": "karma"
    }
}
```

Running npm scripts

There are two types of npm scripts, and the command to run each is slightly different. The first type of npm scripts are "pre-recognized" scripts. These scripts are automatically recognized by npm and don't need a special prefix (as you will see for the other type) to run them. The other type of scripts are "custom" scripts. These scripts aren't pre-recognized by npm and have to be prefixed by a special command to run them. There are a list of pre-recognized scripts in the remarks section.

To run pre-recognized scripts:

`npm start` or `npm test`

To run custom scripts you need to use the `run` command:

`npm run karma`

What npm scripts are and how are they triggered

The npm scripts are commands that `npm` will run for you when called with the proper arguments. The power and sense of this is to NOT install the npm packages globally poluting your environment.

The difference between pre-recognized and custom scripts relies on the `run` word between the tags, **`custom` scripts will need the `run` between npm and the script name**

Based on this we can differenciate and create different tasks or scripts to be run with npm.

Given the following example on the `package.json` file:

```
{
  "name": "MyApp",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "mocha --recursive ./tests/",
    "test:watch": "npm run test -- -w",
    "start": "nodemon --inspect ./app.js",
    "build": "rm -rf ./dist/ && gulp build"
  }
...
}
```

We can see different tasks to be run:

- `npm test` Would work fine as it is a pre-recognized script
- `npm run test` Would work fine as it is a valid way to execute a npm script
- `npm run test:watch` Would work also, and it's calling npm run test inside itself
- `npm run build` Would before running `gulp build` delete the `dist` folder that is in the directory (assuming you are in Linux or the command `rm` is recognized)

Read npm scripts online: https://riptutorial.com/npm/topic/4842/npm-scripts

Chapter 3: Publishing

Examples

Publish public repository

To publish public repositories with a free npm account, you initially need to publish the module with access of public. One way to do this is to set the config for `npm` to read in `packages.json` as follows:

```
"publishConfig": {
  "access": "public"
},
```

you can also use the flag `--access=public` with the `npm publish` command. Otherwise you will get the slightly confusing error message, "you need a paid account to perform this action".

```
npm publish // if you modified packages.json
```

or

```
npm publish --access=public
```

Sign Git Tag

For signing to work you need a default GPG key configured. You can turn it on or off as follows:

```
npm config set sign-git-tag <true or false>
```

Read Publishing online: https://riptutorial.com/npm/topic/3025/publishing

Chapter 4: What NPM does ?

Introduction

It allows you to easily manage different packages (modules) and keep track of which version you've installed.

Examples

Initialize a new NPM project

```
npm init
```

This will initialize a new NPM project for you - just press enter until it stops asking you questions.

Now you'll notice that you have a new file called *package.json*. This file will, among other things, keep track of which packages or modules you've installed in your project.

Read What NPM does ? online: https://riptutorial.com/npm/topic/10586/what-npm-does--

Credits

S. No	Chapters	Contributors
1	Getting started with npm	acdcjunior, Ana, Aurora0001, Cezar Augusto, Community, eppsilon, GilZ, Kenan, Konstantin A. Magg, KriszDev, Paul, Protectator, RationalDev, Tim Anstee
2	npm scripts	Alejandro Vales, DankestMemes
3	Publishing	Pier Paolo Ramon, RationalDev
4	What NPM does ?	Kuhan