npm cheat sheet

For the full **table of contents** see below, but first here is a quick cheatsheet of several `npm` commands:

- Installing npm
- Update npm
- Search for npm packages
- View details of a npm package
- Installing a npm package locally
- Installing a npm package into an application
- Understanding Global versus Local installs in npm
- Global Package Installation
- Uninstalling a package locally
- Uninstalling a package globally
- Installing a specific version of a package
- Cloning a module from Github
- Linking any npm package locally
- Linking local npm packages to multiple applications
- Unlinking a npm package from an application
- Unlinking a npm package from your system
- Create a new npm package
- Creating a new user account on npm
- Publishing a npm package
- Unpublishing a npm package
- Managing owners of packages

Installing npm

back to top

```
curl http://npmjs.org/install.sh | sh
```

Update npm

There are several ways you can update `npm`.

```
curl http://npmjs.org/install.sh | sh
```

or

```
npm install npm -g
```

Search for npm packages

```
npm search hook.io
```

Protip: *Try searching via the browser with **http://browsenpm.org***

View details of a npm package

```
npm view hook.io
```

Installing a npm package locally

back to top

For the purpose of this demo, we will use `http-server`.

`http-server` is a package we've written which provides an easy to use wrapper around node's core http.Server class. This module makes for a good example, since it's API provides both a CLI binary and a requirable node.js module.

```
npm install http-server
```

This performs a **local** install of `http-server` in our **current working directory**

You may also notice a new **`node_modules/`** folder. You can ignore this for now.

Installing a npm package into an application

```
mkdir mynewapp/
cd mynewapp
npm install http-server
touch test.js
```

run script

```
node test.js
```

Notice how we: `require('http-server')`? What kind of wizardry is this?

`http-server` is not the name of a native node.js module. It's the name of the package we just installed from `npm`. `node` and `npm` are smart enough to automatically load modules from our local `node_modules/` folder.

Understanding Global versus Local installs in npm back to top

By default, `npm` will install all packages into the **local** directory you are working in. This is a **good** thing. It can however, be slightly confusing if you have worked with inferior package management systems in the past.

For example, if we:

```
mkdir anotherapp/
cd anotherapp/
touch test.js
```

test.js

```
var HTTPServer = require('http-server');
```

and then run the script...

```
node test.js
```

we'll get this error:

```
node.js:134 throw e; // process.nextTick error, or 'error' event on f
Error: Cannot find module 'http-server'
   at Function._resolveFilename (module.js:326:11)
   at Function._load (module.js:271:25)
   at require (module.js:355:19)
   at Object.<anonymous> (/Users/maraksquires/dev/nodeapps/anotherapp
   at Module._compile (module.js:411:26)
   at Object..js (module.js:417:10)
   at Module.load (module.js:343:31)
   at Function._load (module.js:302:12)
   at Array.<anonymous> (module.js:430:10)
   at EventEmitter._tickCallback (node.js:126:26)
```

This is logical, we installed `http-server` **locally** into `"/mynewapp/"`, **not** in `"/anotherapp/"`.

There are two direct solutions to fix this:

a) Install the package again, but locally into our new application

```
cd anotherapp/
npm install http-server
```

b) Install the package globally

```
npm install http-server -g
```

Global Package Installation

back to top

If you want to have a package available globally use:

```
npm install http-server -g
```

The `-g` flag will indicate that `http-server` should be installed **globally**, and be available for all node scripts to require.

Now, we can `require('http-server')` in any node script on our system.

In addition, since the `http-server` package has specified a `bin` property, it will also install a binary script called `http-server` globally.

Now you can simply run the command:

```
http-server
```

Uninstalling a package locally

```
cd mynewapp/
npm uninstall http-server
```

Uninstalling a package globally

```
npm uninstall http-server -g
```

Installing a specific version of a package

back to top

```
cd mynewapp/
npm install http-server@0.3.0
```

Cloning a module from Github

This is important. In some cases, there will be patches, forks, or branches that we will want to use for our module, but have not yet been published to `npm`. Thankfully, the source code for most `npm` modules is also available on **Github.com**

```
git clone git://github.com/nodeapps/http-server.git
cd http-server/
npm link
```

Our cloned version of `http-server` is now linked locally

Linking any npm package locally

If you have a local directory containing an `npm` package, you can link this package locally. This is good for development purposes and for situations when we do not want to publish our package to the public `npm` repository.

```
cd http-server/
npm link
```

Our local version of `http-server` is "linked" on our local machine

Linking local npm packages to multiple applications

back to top

As we've seen before, `npm` will install packages into the local directory by default. `npm link` works pretty much the same way.

```
mkdir newapp/
cd newapp/
npm link http-server
```

This indicates that we've now linked `http-server` into our new application `newapp`. If we had not run `npm link http-server` we would have gotten a missing module error

Unlinking a npm package from an application

```
cd newapp/
npm unlink http-server
```

Unlinking a npm package from your system

```
cd http-server/
npm unlink
```

Create a new npm package

```
mkdir mypackage/
cd mypackage/
npm init
```

Creating a new user account on npm

```
npm adduser
```

Publishing a npm package

back to top

```
cd mypackage/
npm publish
```

Unpublishing a npm package

```
npm unpublish http-server
```

Managing owners of packages

If you want multiple users to be able to publish to the same package:

```
npm owner add marak http-server
npm owner rm marak http-server
npm owner ls http-server
```

For additional information on the `package.json` format and `npm` best practices, check out Charlie Robbin's article: **http://blog.nodejitsu.com/package-dependencies-done-right**

Table of contents

back to top

PRIVATE

cli
faq
Getting started
web

API

npm bin
npm bugs
npm cache

npm commands	npm config	npm deprecate
npm docs	npm edit	npm explore
npm help search	npm init	npm install
npm link	npm load	npm ls
npm outdated	npm owner	npm pack
npm prefix	npm prune	npm publish
npm rebuild	npm repo	npm restart
npm root	npm run script	npm search
npm shrinkwrap	npm start	npm stop
npm submodule	npm tag	npm test
npm uninstall	npm unpublish	npm update
npm version	npm view	npm whoami
npm		

CLI

npm adduser	npm bin	npm bugs
npm build	npm bundle	npm cache
npm completion	npm config	npm dedupe
npm deprecate	npm docs	npm edit
npm explore	npm help search	npm help
npm init	npm install	npm link
npm ls	npm outdated	npm owner
npm pack	npm prefix	npm prune
npm publish	npm rebuild	npm repo
npm restart	npm rm	npm root
npm run script	npm search	npm shrinkwrap
npm star	npm stars	npm start
npm stop	npm submodule	npm tag
npm test	npm uninstall	npm unpublish
npm update	npm version	npm view
npm whoami	npm	

FILES

npm folders	npmrc	package.json

MISC

npm coding style	npm config	npm developers
npm disputes	npm faq	npm index
npm registry	npm scripts	removing npm
semver		

Nodejitsu.com | Blog | Service Status