[**【原】npm 常用命令详解**](http://www.cnblogs.com/PeunZhang/p/5553574.html)

<http://www.cnblogs.com/PeunZhang/p/5553574.html>

今年上半年在学习gulp的使用，对npm的掌握是必不可少的，经常到[npm官网查询文档](https://docs.npmjs.com/#cli)让我感到不爽，还不如整理了一些常用的命令到自己博客上，于是根据自己的理解简单翻译过来，终于有点输出，想学习npm这块的朋友不可错过这些基础命令，新手出品，对本文有疑惑之处还请同学们不耻下问。很久没有写文章啦，后续也希望自己在nodejs这方面有多点内容产出。



**本文以Windows平台上做测试，以gulp为示例做教程**

# 目录（更新于20160117）

* [npm是什么](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm)
* [npm install 安装模块](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-install)
* [npm uninstall 卸载模块](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-uninstall)
* [npm update 更新模块](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-update)
* [npm outdated 检查模块是否已经过时](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-outdated)
* [npm ls 查看安装的模块](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-ls)
* [npm init 在项目中引导创建一个package.json文件](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-init)
* [npm help 查看某条命令的详细帮助](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-help)
* [npm root 查看包的安装路径](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-root)
* [npm config 管理npm的配置路径](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-config)
* [npm cache 管理模块的缓存](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-cache)
* [npm start 启动模块](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-start)
* [npm stop 停止模块](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-stop)
* [npm restart 重新启动模块](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-restart)
* [npm test 测试模块](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-test)
* [npm version 查看模块版本](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-version)
* [npm view 查看模块的注册信息](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-view)
* [npm adduser  用户登录](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-adduser)
* [npm publish 发布模块](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-publish)
* [npm access 在发布的包上设置访问级别](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-access)
* [npm package.json的语法](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-package.json)

## npm是什么

NPM的全称是Node Package Manager，是随同NodeJS一起安装的包管理和分发工具，它很方便让JavaScript开发者下载、安装、上传以及管理已经安装的包。

## npm install 安装模块

基础语法

[复制代码](javascript:void(0);)

npm install (with no args, in package dir)

npm install [<@scope>/]<name>

npm install [<@scope>/]<name>@<tag>

npm install [<@scope>/]<name>@<version>

npm install [<@scope>/]<name>@<version range>

npm install <tarball file>

npm install <tarball url>

npm install <folder>

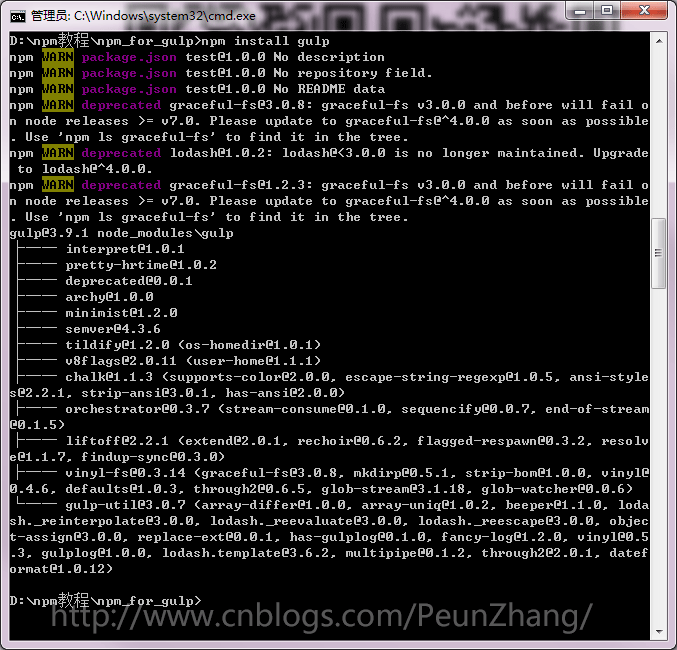
alias: npm i

common options: [-S|--save|-D|--save-dev|-O|--save-optional] [-E|--save-exact] [--dry-run]

[复制代码](javascript:void(0);)

安装包，默认会安装最新的版本

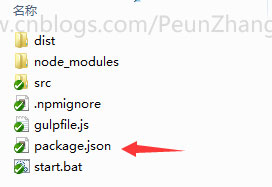
npm install gulp



安装指定版本

npm install gulp@3.9.1

安装包并将信息保持到项目的package.json文件中



项目对模块的依赖可以使用下面的 3 种方法来表示（假设当前版本号是 1.1.0 ）：

* 兼容模块新发布的补丁版本：~1.1.0、1.1.x、1.1
* 兼容模块新发布的小版本、补丁版本：^1.1.0、1.x、1
* 兼容模块新发布的大版本、小版本、补丁版本：\*、x

**-S, --save 安装包信息将加入到dependencies（生产阶段的依赖）**

npm install gulp --save 或 npm install gulp -S

package.json 文件的 dependencies 字段：

"dependencies": {

"gulp": "^3.9.1"

}

**-D, --save-dev 安装包信息将加入到devDependencies（开发阶段的依赖），所以开发阶段一般使用它**

npm install gulp --save-dev 或 npm install gulp -D

package.json 文件的 devDependencies字段：

"devDependencies": {

"gulp": "^3.9.1"

}

**-O, --save-optional 安装包信息将加入到optionalDependencies（可选阶段的依赖）**

npm install gulp --save-optional 或 npm install gulp -O

package.json 文件的optionalDependencies字段：

"optionalDependencies": {

"gulp": "^3.9.1"

}

**-E, --save-exact 精确安装指定模块版本**

npm install gulp --save-exact 或 npm install gulp -E

输入命令**npm install gulp -ES，**留意package.json 文件的 dependencies 字段，以看出版本号中的**^**消失了

"dependencies": {

"gulp": "3.9.1"

}

模块的依赖都被写入了package.json文件后，他人打开项目的根目录（项目开源、内部团队合作），使用npm install命令可以根据dependencies配置安装所有的依赖包

npm install



本地安装（local）

npm install gulp

全局安装（global）,使用 -g 或 --global

npm install gulp -g

## npm uninstall 卸载模块

基础语法

npm uninstall [<@scope>/]<pkg>[@<version>]... [-S|--save|-D|--save-dev|-O|--save-optional]

aliases: remove, rm, r, un, unlink

如卸载开发版本的模块

npm uninstall gulp --save-dev

## npm update 更新模块

基础语法

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

## npm outdated 检查模块是否已经过时

基础语法

npm outdated [[<@scope>/]<pkg> ...]

此命令会列出所有已经过时的包，可以及时进行包的更新



## npm ls 查看安装的模块

基础语法

npm ls [[<@scope>/]<pkg> ...]

aliases: list, la, ll

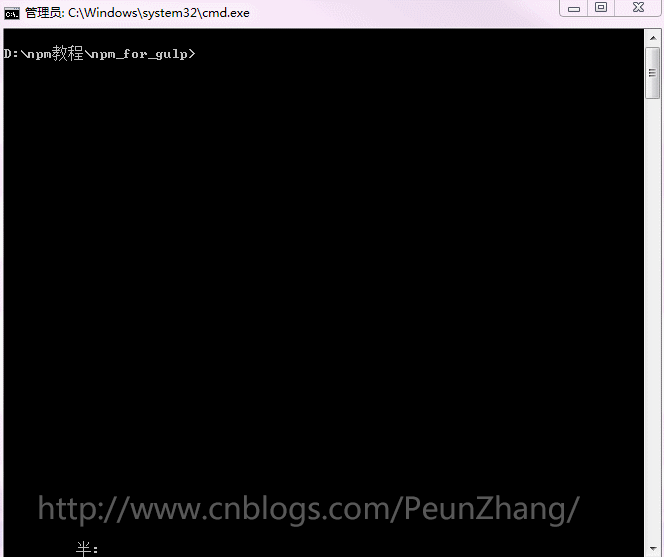
查看全局安装的模块及依赖

npm ls -g

## npm init 在项目中引导创建一个package.json文件

安装包的信息可保持到项目的package.json文件中，以便后续的其它的项目开发或者他人合作使用，也说package.json在项目中是必不可少的。

npm init [-f|--force|-y|--yes]



## npm help 查看某条命令的详细帮助

基础语法

npm help <term> [<terms..>]

例如输入npm help install，系统在默认的浏览器或者默认的编辑器中打开本地nodejs安装包的文件/nodejs/node\_modules/npm/html/doc/cli/npm-install.html

npm help install

## ****npm root****查看包的安装路径

输出 node\_modules的路径

npm root [-g]

## ****npm config****管理npm的配置路径

基础语法

[复制代码](javascript:void(0);)

npm config set <key> <value> [-g|--global]

npm config get <key>

npm config delete <key>

npm config list

npm config edit

npm get <key>

npm set <key> <value> [-g|--global]

[复制代码](javascript:void(0);)

对于config这块用得最多应该是**设置代理，解决npm安装一些模块失败的问题**

例如我在公司内网，因为公司的防火墙原因，无法完成任何模块的安装，这个时候设置代理可以解决

npm config set proxy=http://dev-proxy.oa.com:8080

又如国内的网络环境问题，某官方的IP可能被和谐了，幸好国内有好心人，搭建了镜像，此时我们简单设置镜像

npm config set registry="http://r.cnpmjs.org"

也可以临时配置，如安装淘宝镜像

npm install -g cnpm --registry=https://registry.npm.taobao.org

## npm cache 管理模块的缓存

基础语法

[复制代码](javascript:void(0);)

npm cache add <tarball file>

npm cache add <folder>

npm cache add <tarball url>

npm cache add <name>@<version>

npm cache ls [<path>]

npm cache clean [<path>]

[复制代码](javascript:void(0);)

最常用命令无非清除npm本地缓存

npm cache clean

## npm start 启动模块

基础语法

npm start [-- <args>]

该命令写在package.json文件scripts的start字段中，可以自定义命令来配置一个服务器环境和安装一系列的必要程序，如

"scripts": {

"start": "gulp -ws"

}

此时在cmd中输入npm start命令相当于执行gulpfile.js文件自定义的watch和server命令。

如果package.json文件没有设置start，则将直接启动node server.js

## npm stop 停止模块

基础语法

npm stop [-- <args>]

## npm restart 重新启动模块

基础语法

npm restart [-- <args>]

## npm test 测试模块

基础语法

npm test [-- <args>]

npm tst [-- <args>]

该命令写在package.json文件scripts的test字段中，可以自定义该命令来执行一些操作，如

"scripts": {

"test": "gulp release"

},

此时在cmd中输入npm test命令相当于执行gulpfile.js文件自定义的release命令。

## npm version 查看模块版本

基础语法

[复制代码](javascript:void(0);)

npm version [<newversion> | major | minor | patch | premajor | preminor | prepatch | prerelease | from-git]

'npm [-v | --version]' to print npm version

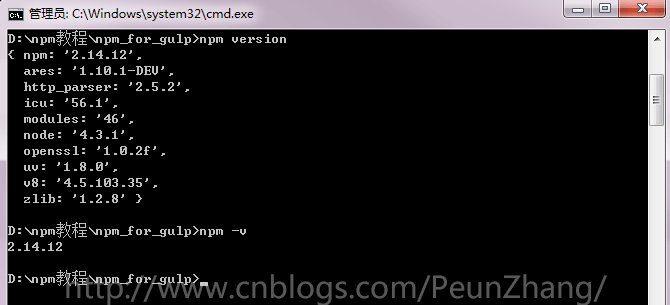
'npm view <pkg> version' to view a package's published version

'npm ls' to inspect current package/dependency versions

[复制代码](javascript:void(0);)

查看模块的版本

npm version



## npm view 查看模块的注册信息

基础语法

npm view [<@scope>/]<name>[@<version>] [<field>[.<subfield>]...]

aliases: info, show, v

查看模块的依赖关系

npm view gulp dependencies

查看模块的源文件地址

npm view gulp repository.url

查看模块的贡献者，包含邮箱地址

npm view npm contributors

## npm adduser 用户登录

基础语法

npm adduser [--registry=url] [--scope=@orgname] [--always-auth]

## http://images2015.cnblogs.com/blog/278431/201701/278431-20170117152013911-489932696.png

发布模板到npm社区前需要先登录，然后再进入发布的操作

## npm publish 发布模块

基础语法

npm publish [<tarball>|<folder>] [--tag <tag>] [--access <public|restricted>]

Publishes '.' if no argument supplied

Sets tag 'latest' if no --tag specified

## http://images2015.cnblogs.com/blog/278431/201701/278431-20170117152224005-1557216239.png

## npm access 在发布的包上设置访问级别

基础语法

[复制代码](javascript:void(0);)

npm access public [<package>]

npm access restricted [<package>]

npm access grant <read-only|read-write> <scope:team> [<package>]

npm access revoke <scope:team> [<package>]

npm access ls-packages [<user>|<scope>|<scope:team>]

npm access ls-collaborators [<package> [<user>]]

npm access edit [<package>]

[复制代码](javascript:void(0);)

## npm package.json的语法

英文原版：<https://docs.npmjs.com/files/package.json>

这块内容好多，国内有好心人整理：《[npm的package.json中文文档](https://github.com/ericdum/mujiang.info/issues/6/)》，从这份文档拷贝出一些比较常见的，如下：

**默认值**

npm会根据包内容设置一些默认值。

* "scripts": {"start": "node server.js"}

如果包的根目录有server.js文件，npm会默认将start命令设置为node server.js。

* "scripts":{"preinstall": "node-waf clean || true; node-waf configure build"}

如果包的根目录有wscript文件，npm会默认将preinstall命令用node-waf进行编译。

* "scripts":{"preinstall": "node-gyp rebuild"}

如果包的根目录有binding.gyp文件，npm会默认将preinstall命令用node-gyp进行编译。

* "contributors": [...]

如果包的根目录有AUTHORS文件，npm会默认逐行按Name <email> (url)格式处理，邮箱和url是可选的。#号和空格开头的行会被忽略。

**name**

在package.json中最重要的就是name和version字段。他们都是必须的，如果没有就无法install。name和version一起组成的标识在假设中是唯一的。改变包应该同时改变version。

name是这个东西的名字。注意：

* 不要把node或者js放在名字中。因为你写了package.json它就被假定成为了js，不过你可以用"engine"字段指定一个引擎（见后文）。
* 这个名字会作为在URL的一部分、命令行的参数或者文件夹的名字。任何non-url-safe的字符都是不能用的。
* 这个名字可能会作为参数被传入require()，所以它应该比较短，但也要意义清晰。
* 在你爱上你的名字之前，你可能要去npm registry查看一下这个名字是否已经被使用了。<http://registry.npmjs.org/>

**version**

version必须能被[node-semver](https://github.com/isaacs/node-semver)解析，它被包在npm的依赖中。（要自己用可以执行npm install semver）

可用的“数字”或者“范围”见[semver(7)](https://npmjs.org/doc/misc/semver.html).

**description**

放简介，字符串，方便在npm search中搜索

**keywords**

关键字，数组、字符串，方便在npm search中搜索

**bugs**

你项目的提交问题的url和（或）邮件地址

{

"url" : "http://github.com/owner/project/issues",

"email" : "project@hostname.com"

}

**license**

你应该要指定一个许可证，让人知道使用的权利和限制的。

最简单的方法是，假如你用一个像BSD或者MIT这样通用的许可证，就只需要指定一个许可证的名字，像这样：

{ "license" : "BSD" }

如果你又更复杂的许可条件，或者想要提供给更多地细节，可以这样:

"licenses" : [

{ "type" : "MyLicense"

, "url" : "http://github.com/owner/project/path/to/license"

}

]

**repository**

指定你的代码存放的地方。这个对希望贡献的人有帮助。如果git仓库在github上，那么npm docs命令能找到你。

这样做：

[复制代码](javascript:void(0);)

"repository" :

{ "type" : "git"

, "url" : "http://github.com/isaacs/npm.git"

}

"repository" :

{ "type" : "svn"

, "url" : "http://v8.googlecode.com/svn/trunk/"

}

[复制代码](javascript:void(0);)

URL应该是公开的（即便是只读的）能直接被未经过修改的版本控制程序处理的url。不应该是一个html的项目页面。因为它是给计算机看的。

**scripts**

“scripts”是一个由脚本命令组成的hash对象，他们在包不同的生命周期中被执行。key是生命周期事件，value是要运行的命令。

参考上面的[npm start](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-start)、[npm test](http://www.cnblogs.com/PeunZhang/p/5553574.html#npm-test)命令

更多详细请看 [npm-scripts(7)](https://npmjs.org/doc/misc/npm-scripts.html)

**config**

"config" hash可以用来配置用于包脚本中的跨版本参数。在实例中，如果一个包有下面的配置：

{

"name" : "foo",

"config" : { "port" : "8080" }

}

然后有一个“start”命令引用了npm\_package\_config\_port环境变量，用户可以通过npm config set foo:port 8001来重写他。

参见 [npm-config(7)](https://npmjs.org/doc/misc/npm-config.html) 和 [npm-scripts(7)](https://npmjs.org/doc/misc/npm-scripts.html)。

**dependencies**

依赖是给一组包名指定版本范围的一个hash。这个版本范围是一个由一个或多个空格分隔的字符串。依赖还可以用tarball或者git URL。

请不要将测试或过渡性的依赖放在dependencieshash中。见下文的devDependencies

**详见**[**semver(7)**](https://npmjs.org/doc/misc/semver.html)**.**

* version 必须完全和version一致
* >version 必须比version大
* >=version 同上
* <version 同上
* <=version 同上
* ~version 大约一样，见[semver(7)](https://npmjs.org/doc/misc/semver.html)
* 1.2.x 1.2.0, 1.2.1, 等，但不包括1.3.0
* http://... 见下文'依赖URL'
* \* 所有
* "" 空，同\*
* version1 - version2 同 >=version1 <=version2.
* range1 || range2 二选一。
* git... 见下文'依赖Git URL'
* user/repo 见下文'GitHub URLs'

比如下面都是合法的：

[复制代码](javascript:void(0);)

{ "dependencies" :

{ "foo" : "1.0.0 - 2.9999.9999"

, "bar" : ">=1.0.2 <2.1.2"

, "baz" : ">1.0.2 <=2.3.4"

, "boo" : "2.0.1"

, "qux" : "<1.0.0 || >=2.3.1 <2.4.5 || >=2.5.2 <3.0.0"

, "asd" : "http://asdf.com/asdf.tar.gz"

, "til" : "~1.2"

, "elf" : "~1.2.3"

, "two" : "2.x"

, "thr" : "3.3.x"

}

}

[复制代码](javascript:void(0);)

**devDependencies**

如果有人要使用你的模块，那么他们可能不需要你开发使用的外部测试或者文档框架。

在这种情况下，最好将这些附属的项目列在devDependencies中。

这些东西会在执行npm link或者npm install的时候初始化，并可以像其他npm配置参数一样管理。详见[npm-config(7)](https://npmjs.org/doc/misc/npm-config.html)。

对于非特定平台的构建步骤，比如需要编译CoffeeScript，可以用prepublish脚本去实现，并把它依赖的包放在devDependency中。（译者注：prepublish定义了在执行npm publish的时候先行执行的脚本）

比如：

[复制代码](javascript:void(0);)

{ "name": "ethopia-waza",

"description": "a delightfully fruity coffee varietal",

"version": "1.2.3",

"devDependencies": {

"coffee-script": "~1.6.3"

},

"scripts": {

"prepublish": "coffee -o lib/ -c src/waza.coffee"

},

"main": "lib/waza.js"

}

[复制代码](javascript:void(0);)

prepublish脚本会在publishing前运行，这样用户就不用自己去require来编译就能使用。并且在开发模式中（比如本地运行npm install）会运行这个脚本以便更好地测试。

参考资料：

<https://docs.npmjs.com/#cli>

<https://github.com/ericdum/mujiang.info/issues/6/>

<https://segmentfault.com/a/1190000004221514>

赶在高考和端午节前的一篇文章，祝高考的同学们超常发挥，也祝大家粽子节快乐，多多陪陪家人。

**作者：白树**

**出处：**[**http://peunzhang.cnblogs.com/**](http://peunzhang.cnblogs.com/)