npm使用入门

宁静的夜 关注
0.1 2017.03.10 18:31* 字数 1778 阅读 19238 评论 0 喜欢 24 赞赏 1

NPM是什么

npm

NPM能做什么?

npm的安装、卸载、升级、配置
npm的使用：package的安装、卸载、升级、查看、搜索、发布
npm包的安装模式：本地 vs 全局
package.json：包描述信息
package版本：常见版本声明形式

> npm makes it easy for JavaScript developers to share and reuse code, and it makes it easy to update the code that you're sharing.

简单来说，npm就是javascript的包管理工具，类似java语法当中的maven, gradle, python的pip。

安装

npm是和Node.js一起发布的，只要安装了Node.js，npm也安装好了，可以从Node.js的下载页下载对应操作系统的安装包安装即可。 安装好后，执行如下命令，检查是否安装成功。

```
$ node -v
v6.10.0

$ npm -v
4.4.1
```

但是由于npm自身的更新频率比Node.js高很多，所以通过上面的命令安装的npm可能不是最新版本，可以通过下面的命令单独更新npm

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

安装包

执行命令

```
$ npm install <package_name>
```

便可以安装对应的包到执行命令的当前目录，并创建一个`node_modules`的文件夹,然后把需要安装的安装包下载到里面。

使用`package.json`

通过上面的命令，直接安装的包默认都是最新版本的。但是在项目中，我们怎么让一起开发的同事知道项目中用了哪些包，具体包的版本信息呢？这时`package.json`就上场了，可以把它想成是java语言中的`pom.xml`，
python语言中的`requirements.txt`。

一个基本的`package.json`文件至少需要包含两个重要信息: 包名`name`和版本信息`version`。
例如:

```
{
  "name": "my-awesome-package",
  "version": "1.0.0"
}
```

创建`package.json`

我们可以使用命令`npm init`来初始化一个`package.json`文件，运行这个命令后，它会询问一些关于包的基本信息，根据实际情况回答即可。如果不喜欢这种方式，可以使用`npm init --yes`命令直接使用默认的配置来创建`package.json`文件，最后根据需要修改创建好的`package.json`文件即可。

`package.json`文件创建好后，我们来看看它长得什么样子吧!

```
{
  "name": "my_package",
  "description": "",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "https://github.com/ashleygwilliams/my_package.git"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/ashleygwilliams/my_package/issues"
  },
  "homepage": "https://github.com/ashleygwilliams/my_package"
}
```

主要字段的含义如下:

- name: 模块名, 模块的名称有如下要求:
  - 全部小写
  - 只能是一个词语，没有空格
  - 允许使用破折号和下划线作为单词分隔符
- version: 模块版本信息
- description：关于模块功能的简单描述，如果这个字段为空的话，默认会从当前目录的`READMD.md`或`README`文件读取第一行内容作为它的默认值。
- main: 模块被引入后，首先加载的文件，默认为`index.js`。
- scripts: 定义一些常用命令入口

类似git一样，npm也可以做一些简单的配置来设置一些我们常用的信息

```
$ npm set init.author.email "wombat@npmjs.com"

$ npm set init.author.name "ag_dubs"

$ npm set init.license "MIT"
```

这样下次执行`npm init`的时候，就会用上我们配置的一些默认信息啦!

搜索模块

package搜索

```
1  $ npm search grunt-cli
```

返回结果如下

```
npm http GET http://registry.npmjs.org/-/all/since?stale=update_after&startkey=1375519407838
npm http 200 http://registry.npmjs.org/-/all/since?stale=update_after&startkey=1375519407838
NAME DESCRIPTION AUTHOR DATE KEYWORDS
grunt-cli The grunt command line interface. =cowboy =tkellen 2013-07-27 02:24
grunt-cli-dev-exitprocess The grunt command line interface. =dnevnik 2013-03-11 16:19
grunt-client-compiler Grunt wrapper for client-compiler. =rubenv 2013-03-26 09:15 gruntplugin
grunt-clientside Generate clientside js code from CommonJS modules =jga 2012-11-07 01:20 gruntplugin
```

安装模块

如果我们的项目依赖了很多package，一个一个地安装那将是个体力活。
我们可以将项目依赖的包都在package.json这个文件里声明，然后一行命令搞定

使用 `npm install` 会读取 `package.json` 文件来安装模块。安装的模块分为两类 `dependencies` 和 `devDependencies`，分别对应生产环境需要的安装包和开发环境需要的安装包。

同样在安装模块的时候，可以通过指定参数来修改 `package.json` 文件，如 `npm install --help`

```
$ npm install <package_name> --save

$ npm install <package_name> --save-dev
```

来将新安装的模块信息记录到 `package.json` 文件。

更新模块

```
$ npm update
```

卸载模块

```
$ npm uninstall <package_name>
```

如果要在卸载模块的同时，也将他从package.json文件中移除，可以添加跟安装时候一样的参数，例如:

```
$ npm uninstall --save lodash
```

查看模块

```
$ npm ls
```

如果是要查看package的全局安装信息，加上 `-g` 就可以。

查看特定package的信息

运行如下命令，输出grunt-cli的信息

```
1  $ npm ls grunt-cli
```

输出的信息比较有限，只有安装目录、版本，如下:

```
/private/tmp/npm
└── grunt-cli@0.1.9
```

如果要查看更详细信息，可以通过 `npm info pkg`，输出的信息非常详尽，包括作者、版本、依赖等。

```
1  $ npm info grunt-cli
```

默认情况下。我们执行默认的安装命令安装的包都是安装到当前目录下的，只能在当前目录下使用。但是假如我们需要使用一些全局的软件，如 `grunt`，我们可以在安装的时候，添加 `-g` 选项来安装，方便后面在任何目录下都可以使用 `grunt` 相关的命令

```
$ npm install -g grunt
```

同理，更新全局的安装包只需要执行命令

```
$ npm update -g
```

为了查看当前哪些包需要更新，可以使用如下命令来查看

```
$ npm outdated -g --depth=0
Package             Current  Wanted  Latest  Location
vue-cli               2.7.0   2.8.1   2.8.1
webpack              1.13.2   2.2.1   2.2.1
webpack-dev-server   1.14.1   2.4.1   2.4.1
```

卸载全局安装的包也只需要加上 `-g` 选项即可。如

```
$ npm uninstall -g jshint
```

创建自己的Node.js模块

一个Node.js模块就是一个可以发布到npm，供其他开发者下载和使用的模块。那么，到底怎样和其他开发者分享我们的模块呢?

首先，我们必须创建一个 `package.json` 文件，添加上关于我们想要分享的模块信息，如：模块功能，开发者信息等。

一旦 `package.json` 文件创建好后，我们需要创建一个模块被引入时，就加载的文件。
即 `package.json` 中 `main` 字段指定的文件，默认为 `index.js`。我们需要在文件中将一个函数赋值给 `exports` 模块，方便其他开发者调用我们的模块。如

```
exports.printMsg = function() {
  console.log("This is a message from the demo package");
}
```

包(Pacakges)和模块(Modules)

在使用npm的时候，有两个概念容易搞混，那就是包(Pacakges)和模块(Modules)。简单来说，包和模块的区别如下：

- 包是一个被 `package.json` 文件描述了的文件或者目录
- 模块是可以被Node.js引用的文件或目录

什么是包?

> a) a folder containing a program described by a package.json file
> b) a gzipped tarball containing (a)
> c) a url that resolves to (b)
> d) a <name>@<version> that is published on the registry with (c)
> e) a <name>@<tag> that points to (d)
> f) a <name> that has a latest tag satisfying (e)
> g) a git url that, when cloned, results in (a).

什么是模块?

> - A folder with a package.json file containing a main field.
> - A folder with an index.js file in it.
> - A JavaScript file.

npm配置

在使用npm时，我们可以根据个人的需要，指定很多配置信息。
npm的配置信息加载优先级如下(从高到低)

1. 命令行参数
2. 环境变量
3. `npmrc` 文件
    1. 项目级别的npmrc文件(/path/to/my/project/.npmrc)
    2. 用户级别的npmrc文件（~/.npmrc)

3. 全局的npmrc文件($PREFIX/etc/npmrc)
4. npm内置的npmrc文件(/path/to/npm/npmrc)

查看配置

```
$ npm config list -l
```

配置npm源

最后介绍一个比较重要的配置，当我们使用默认配置从npm官网下载模块时，由于网络的因素，会导致我们的下载速度特别慢。所以，我们可以配置一些国内的镜像来加快我们的下载速度。在这里，我推荐使用[淘宝的npm镜像](淘宝 NPM 镜像), 具体使用方式如下:

- 临时使用, 安装包的时候通过 `--registry` 参数即可

```
$ npm install express --registry https://registry.npm.taobao.org
```

- 全局使用

```
$ npm config set registry https://registry.npm.taobao.org
// 配置后可通过下面方式来验证是否成功
npm config get registry
// 或
npm info express
```

- 使用cnpm使用

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

// 使用cnpm安装包
cnpm install express
```

更多使用

请参考官方文档

NPM配置

npm的配置工作主要是通过 npm config 命令，主要包含增、删、改、查几个步骤，下面就以最为常用的proxy配置为例。

设置proxy

内网使用npm很头痛的一个问题就是代理，假设我们的代理是 http://proxy.example.com:8080，那么命令如下：

```
$ npm config set proxy http://proxy.example.com:8080
```

由于 npm config set 命令比较常用，于是可以如下简写

```
$ npm set proxy http://proxy.example.com:8080
```

查看proxy

设置完，我们查看下当前代理设置

```
$ npm config get proxy
```

输出如下：

> http://proxy.example.com:8080/

同样可如下简写：

```
$ npm get proxy
```

删除proxy

代理不需要用到了，那删了吧

```
$ npm delete proxy
```

查看所有配置

```
$ npm config list
```

修改配置文件

有时候觉得一条配置一条配置地修改有些麻烦，就直接进配置文件修改了

```
$ npm config edit
```

内容只是简单地把最常见的命令，以及一些需要了解的内容列了出来。如要进一步了解，可参考官网说明。此外，npm help 是我们最好的朋友，如果忘了有哪些命令，命令下有哪些参数，可通过help进行查看。