第1章 NPM 包管理工具

1.1 什么是 NPM

NPM 全称 Node Package Manager，它是 JavaScript 的包管理工具, 并且是 Node.js 平台的默认包管理工具。通过 NPM 可以安装、共享、分发代码,管理项目依赖关系。

- 可从NPM服务器下载别人编写的第三方包到本地使用。
- 可从NPM服务器下载并安装别人编写的命令行程序到本地使用。
- 可将自己编写的包或命令行程序上传到NPM服务器供别人使用。

其实我们可以把 NPM 理解为前端的 Maven。我们通过 npm 可以很方便地安装与下载 js 库，管理前端工程。

最新版本的 Node.js 已经集成了 npm 工具，所以必须首先在本机安装 Node 环境。

Node.js 官网下载地址：

- 英文网：https://nodejs.org/en/download/
- 中文网：http://nodejs.cn/download/

装完成后，查看当前 nodejs 与 npm 版本

```
C:\Users\Administrator>node -v
v10.15.3

C:\Users\Administrator>npm -v
6.4.1
```

1.2 NPM 初始化项目

`npm init` 命令初始化项目：

- 新建一个 `npm-demo` 文件夹，通过命令提示符窗口进入到该文件夹，执行下面命令进行初始化项目

```
npm init
```

根据提示输入相关信息，如果使用默认值，则直接回车即可。

  - package name: 包名，其实就是项目名称，注意不能有大写字母
- version: 项目版本号
  - description: 项目描述
  - keywords: {Array}关键字，便于用户搜索到我们的项目

```
D:\StudentProject\WebStudy\npm-demo>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (npm-demo) npmDemo   包名不能有大写字母
Sorry, name cannot contain leading or trailing spaces and name can only contain
URL-friendly characters and name can no longer contain capital letters.
package name: (npm-demo)   按回车，取（）括号中的默认值
version: (1.0.0)
description: first npm demo
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to D:\StudentProject\WebStudy\npm-demo\package.json:

                                   会生成一个package.json保存位置
{
  "name": "npm-demo",
  "version": "1.0.0",
  "description": "first npm demo",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

- 最后会生成 `package.json` 文件，这个是包的配置文件，相当于 maven 的 pom.xml 我们之后也可以根据需要进行修改。

```
{
    "name": "npm-demo", //包名
    "version": "1.0.0", //版本号
    "description": "first npm demo", // 描述
    "main": "index.js", //程序的主入口文件 index.js
    "scripts": { // 脚本命令组成的对象，如果 test 测试环境，dev 开发环境， prod 生产环境
        "test": "echo \"Error: no test specified\" && exit 1"
    },
    "author": "",
    "license": "ISC" //许可证 默认即可
}
```

- 初始化项目，均采用默认信息，不会提示你手动输入信息

```
npm init -y
```

1.3 安装模块

1.3.1 安装方式

`npm install` 命令用于安装某个模块，安装方式分为 ：本地安装（local）、全局安装（global）两种.

- **本地安装**

  将 JS 库安装在**当前执行命令时所在目录下**

```
# 本地安装命令，版本号可选
npm install <Module Name>[@版本号]
```

- **全局安装**

  将 JS 库安装到你的`全局目录`下

```
# 全局安装命令
npm install <Module Name>[@版本号] -g
```

- 如果安装时出现如下错误：

  `npm err! Error: connect ECONNREFUSED 127.0.0.1:8087`

  解决方法，执行如下命令：

  `npm config set proxy null`

1.3.2 本地安装

本地安装会将 js 库安装在当前目录下

- **安装最新版`express`模块，**它是基于Node.js 平台的 Web 开发框架，执行如下命令：

```
npm install express
```

  - 如果出现黄色的是警告信息，可以忽略，请放心，你已经成功安装了。
  - 在该目录下会出现一个 `node_modules` 文件夹 和 `package-lock.json`
    - `node_modules` 文件夹用于存放下载的js库（相当于maven的本地仓库）
    - `package-lock.json` 是在 `npm install` 时候生成一份文件。

      用以记录当前状态下实际安装的各个包的具体来源和版本号。
    - 重新打开 `package.json` 文件，发现刚才下载的 jquery.js 已经添加到依赖列表中了.

```
{
  "name": "npm-demo",
  "version": "1.0.0",
  "description": "first npm project",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "jquery": "^3.4.0"
  }
}
```

- 关于模块版本号表示方式：
  - 指定版本号：比如 `3.5.2`，只安装指定版本。遵循 “大版本.次要版本.小版本”的格式规定。
  - ~波浪号 + 指定版本号：比如 `~3.5.2`，安装 3.5.x 的最新版本（不低于 3.5.2），但是不安装 3.6.x，也就是说安装时不改变大版本号和次要版本号。
  - ^ 插入号 + 指定版本号：比如 `^3.5.2`，安装 3.x.x 的最新版本（不低于 3.5.2），但是不安装 4.x.x，也就是说安装时不改变大版本号。需要注意的是，如果大版本号为0，则插入号的行为与波浪号相同，这是因为此时处于开发阶段，即使是次要版本号变动，也可能带来 程序的不兼容。
  - `latest`：安装最新版本。

- **安装指定版本模块**

  安装 jquery 2.2.0版本的jquery

```
npm install jquery@2.2.0
```

1.3.3 全局安装

使用全局安装会将库安装到你的全局目录下。

查看全局安装目录

- 如果你不知道你的全局目录在哪里，执行命令:

```
npm root -g
```

```
D:\StudentProject\WebStudy\npm-demo>npm root -g
C:\Users\Administrator\AppData\Roaming\npm\node_modules\node_modules
```

我的全局目录是：`C:\Users\Administrator\AppData\Roaming\npm\node_modules`

- 修改默认全局安装目录，使用命令 ：

  比如我修改为 `D:\npm`

```
npm config set prefix "D:\npm"
```

```
D:\StudentProject\WebStudy\npm-demo>npm root -g
C:\Users\Administrator\AppData\Roaming\npm\node_modules\node_modules 默认路径

D:\StudentProject\WebStudy\npm-demo>npm config set prefix "D:\npm" 修改全局路径

D:\StudentProject\WebStudy\npm-demo>npm root -g
D:\npm\node_modules  修改后的全局路径

D:\StudentProject\WebStudy\npm-demo>
```

全局安装

- **全局安装 vue 模块，执行命令**

```
npm install vue -g
```

```
D:\StudentProject\WebStudy\npm-demo>npm install vue -g
+ vue@2.6.10
added 1 package from 1 contributor in 1.546s
```

- 查看全局已安装模块：

```
npm list -g
```

```
D:\StudentProject\WebStudy\npm-demo>npm list -g
D:\npm
`-- vue@2.6.10


D:\StudentProject\WebStudy\npm-demo>
```

1.3.4 生产环境模块安装

- **格式：**

`--save` 或 `-S` 参数意思是把模块的版本信息保存 `package.json` 文件的 `dependencies` 字段中（生产环境依赖）

```
npm install <Module Name> [--save|-S]
```

- **举例：**

  安装 vue 模块，安装在生产环境依赖中

```
npm install vue -S
```

在 `package.json` 文件的 `dependencies` 字段中

```
"dependencies": {
    "express": "^4.17.0",
    "jquery": "^2.2.0",
    "vue": "^2.6.10"
}
```

1.3.5 开发环境模块安装

- **格式：**

  `--save-dev` 或 `-D` 参数是把模块版本信息保存到 `package.json` 文件的 `devDependencies` 字段中（开发环境依赖),所以开发阶段一般使用它:

```
npm install <Module Name> [--save-dev|-D]
```

- **举例：**

  安装 eslint 模块，它是语法格式校验，只在开发环境依赖中即可

```
npm install eslint -D
```

在 package.json 文件的 `devDependencies` 字段中

```
"devDependencies": {
    "eslint": "^5.16.0"
}
```

1.4 批量下载模块

我们从网上下载某些项目后，发现只有 `package.json`，没有 `node_modules` 文件夹，这时我们需要通过命令下载这些js库。

- 命令提示符进入 `package.json` 所在目录，执行命令：

```
npm install
```

此时，npm 会自动下载 `package.json` 中依赖的js库.

1.5 查看模块命令

1.5.1 查看本地已安装模块方式

- 方式1：可以安装目录 `node_modules` 下的查看包是否还存在
- 方式2：可以使用以下命令查看：

```
# 查看本地安装的所有模块
npm list
# 查看指定模板
npm list <Module Name>
```

1.5.2 查看模块远程最新版本

1. 格式

```
npm view <Module Name> version
```

2. 举例: 查看 jquery 模块的最新版本

```
npm view jquery version
```

1.5.3 查看模块远程所有版本

1. 格式

```
npm view <Module Name> versions
```

2. 举例: 查看 jquery 模块的所有版本

```
npm view jquery versions
```

1.6 卸载模块

- 卸载局部模块

```
npm uninstall <Module Name>
```

- 卸载全局模块

```
npm uninstall -g <Module Name>
```

1.7 配置淘宝镜像加速

1. 查看当前使用的镜像地址

```
npm get registry
```

2. **配置淘宝镜像地址**

```
npm config set registry https://registry.npm.taobao.org
```

3. 安装下载模块

```
npm install <Module Name>
```

4. 还原默认镜像地址

```
npm config set registry https://registry.npmjs.org/
```