Node.js npm

关注者 106 被浏览 48,979

# npm install 生成的package-lock.json是什么文件？有什么用？

前段时间升级了Node.js，现在执行`npm install`的时候，就会在当前目录生成一个`package-lock.json`的文件。 但不知道这...显示全部

关注问题 写回答 添加评论 分享 邀请回答 举报 ···

## 5 个回答

默认排序

**二口南洋**
有赞招前端/lvdada@youzan.com

72 人赞同了该回答

2018.3.1更 最近入职了杭州有赞，我们组（微商城）上半年的前端hc有8个，缺人缺人缺人！恳请各路大佬发简历！ lvdada#youzan.com

---

上面的答案把 package-lock.json 的作用写的很明确了，我来说说不一样的点。

npm 5.x 发布以来到现在的5.6.0 lock的作用变更过好多次，现在网上很多小白文都是停留在以前的文档翻译。

从npm3.x更新到了npm5，但是发现执行 `npm i ` 时的现象跟网络上的科普文不太一致。

有提到不管怎么修改package.json文件，重复执行npm i，npm都会根据lock文件描述的版本信息进行下载。

也有提到重复npm i时，npm会不顾lock的信息，根据package.json中的包Semantic versioning 版本信息下载更新模块（lock貌似没啥用了）。

**查阅资料得知，自npm 5.0版本发布以来，npm i的规则发生了三次变化。**

1、npm 5.0.x 版本，不管package.json怎么变，npm i 时都会根据lock文件下载

package-lock.json file not updated after package.json file is changed · Issue #16866 · npm/npm

这个 issue 控诉了这个问题，明明手动改了package.json，为啥不给我升级包！然后就导致了5.1.0的问题...

2、5.1.0版本后 npm install 会无视lock文件 去下载最新的npm

然后有人提了这个issue why is package-lock being ignored? · Issue #17979 · npm/npm

控诉这个问题，最后演变成5.4.2版本后的规则。

## 相关问题

世界上还有哪些比 npm 更坏的包管理器？ 21 个回答

npm install的实现原理？ 7 个回答

package-lock.json 需要写进 .gitignore 吗？ 9 个回答

npm 5 发布，有什么值得关注的新特性吗？ 3 个回答

为什么 npm 要为每个项目单独安装一遍 node_modules？ 20 个回答

## 相关推荐

画龙点睛——从Windows程序到商业产品
★★★★★ 37 人参与

60 分钟学会 Lightroom 修图基础
★★★★★ 2132 人参与


跟老男孩学 Linux 运维：核心系统命令实战
1,096 人读过 阅读

3、5.4.2版本后 why is package-lock being ignored? · Issue #17979 · npm/npm

大致意思是，如果改了package.json，且package.json和lock文件不同，那么执行`npm i`时npm会根据package中的版本号以及语义含义去下载最新的包，并更新至lock。

如果两者是同一状态，那么执行`npm i `都会根据lock下载，不会理会package实际包的版本是否有新。

编辑于 2018-03-03

**胡夏**
爱音乐和滑板的IT人。

26 人赞同了该回答

`package.json` 里面定义的是版本范围（比如 `^1.0.0` ），具体跑 `npm install` 的时候安的什么版本，要解析后才能决定，这里面定义的依赖关系树，可以称之为逻辑树（logical tree）。

`node_modules` 文件夹下才是npm实际安装的确定版本的东西，这里面的文件夹结构我们可以称之为物理树（physical tree）。

安装过程中有一些去重算法，所以你会发现逻辑树结构和物理树结构不完全一样。

`package-lock.json` 可以理解成对结合了逻辑树和物理树的一个快照（snapshot），里面有明确的各依赖版本号，实际安装的结构，也有逻辑树的结构。

其最大的好处就是能获得可重复的构建（repeatable build），当你在CI（持续集成）上重复build的时候，得到的artifact是一样的，因为依赖的版本都被锁住了。在npm5以后，其内容和 `npm-shrinkwrap.json` 一模一样。

这些在npm官网文档都有详细解释。

编辑于 2017-11-11

▲ 26 ▼ 2 条评论 分享 收藏 感谢 ···

**六点大**
打酱油

12 人赞同了该回答

自问自答：

> **package-lock.json** is automatically generated for any operations where npm modifies either the node_modules tree, or **package.json**. It describes the exact tree that was generated, such that subsequent installs are able to generate identical trees, regardless of intermediate dependency updates.

大概意思好像是：

**package-lock.json**是当 node_modules 或 package.json 发生变化时自动生成的文件。这个文件主要功能是确定当前安装的包的依赖，以便后续重新安装的时候生成相同的依赖，而忽略项目开发过程中有些依赖已经发生的更新。

对比之下，大概是想做类似 Yarn 的功能。

详细内容，请自行查阅官方文档：docs.npmjs.com/files/pa...

编辑于 2017-07-13

▲ 12 ▼ 5 条评论 分享 收藏 感谢 ···

**周载南**
谷歌软件工程师

32 人赞同了该回答

根据官方文档，这个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npm package的具体来源和版本号。

它有什么用呢？因为npm是一个用于管理package之间依赖关系的管理器，它允许开发者在pacakge.json中间标出自己项目对npm各库包的依赖。你可以选择以如下方式来标明自己所需要库包的版本

这里举个例子：

```
"dependencies": {
  "@types/node": "^8.0.33",
},
```

这里面的 向上标号^是定义了**向后（新）兼容依赖**，指如果 types/node的版本是超过8.0.33，并在大版本号（8）上相同，就允许下载最新版本的 types/node库包，例如实际上可能运行npm install时候下载的具体版本是8.0.35。波浪号

大多数情况这种向新兼容依赖下载最新库包的时候都没有问题，可是因为npm是开源世界，各库包的版本语义可能并不相同，有的库包开发者并不遵守严格这一原则：相同大版本号的同一个库包，其接口符合兼容要求。这时候用户就很头疼了：在完全相同的一个nodejs的代码库，在不同时间或者不同npm下载源之下，下到的各依赖库包版本可能有所不同，因此其依赖库包行为特征也不同有时候甚至完全不兼容。

因此npm最新的版本就开始提供自动生成package-lock.json功能，为的是让开发者知道只要你保存了源文件，到一个新的机器上、或者新的下载源，只要按照这个package-lock.json所标示的具体版本下载依赖库包，就能确保所有库包与你上次安装的完全一样。

编辑于 2017-10-30

▲ 32 ▼ 1 条评论 分享 收藏 感谢 ···

Mars Wong
偷渡中的码农

npm 的依赖版本管理非常宽松，同一个项目一天之内可能会 install 到不同版本的依赖，依赖包的升级有可能会给你的项目带来 bug，这时候 shrinkwrap 应运而生了，但是，几乎所有人都懒得写，所以，在受到 yarn.lock 的强烈冲击之后，npm 还是决定在 5.0.0+ 版本默认生成 package-lock.json，就是说你现在不需要写 npm-shrinkwrap.json 了

发布于 2017-07-12

▲ 0 ▼ 添加评论 分享 收藏 感谢 ···

写回答

2 个回答被折叠（为什么？）