package-lock.json的作用

其实用一句话来概括很简单，就是锁定安装时的包的版本号，并且需要上传到git，以保证其他人在npm install时大家的依赖能保证一致。

引用知乎@周载南的回答

> 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。

原来package.json文件只能锁定大版本，也就是版本号的第一位，并不能锁定后面的小版本，你每次npm install都是拉取的该大版本下的最新的版本，为了稳定性考虑我们几乎是不敢随意升级依赖包的，这将导致多出来很多工作量，测试/适配等，所以package-lock.json文件出来了，当你每次安装一个依赖的时候就锁定在你安装的这个版本。

那如果我们安装时的包有bug，后面需要更新怎么办?

在以前可能就是直接改package.json里面的版本，然后再npm install了，但是5版本后就不支持这样做了，因为版本已经锁定在package-lock.json里了，所以我们只能npm install xxx@x.x.x 这样去更新我们的依赖，然后package-lock.json也能随之更新。

假如我已经安装了jquery 2.1.4这个版本，从git更新了package.json和package-lock.json，我npm install能覆盖掉node_modules里面的依赖吗?

其实我也有这个疑问，所以做了测试，在直接更新package.json和package-loc.json这两个文件后，npm install是可以直接覆盖掉原先的版本的，所以在协作开发时，这两个文件如果有更新，你的开发环境应该npm install一下才对。