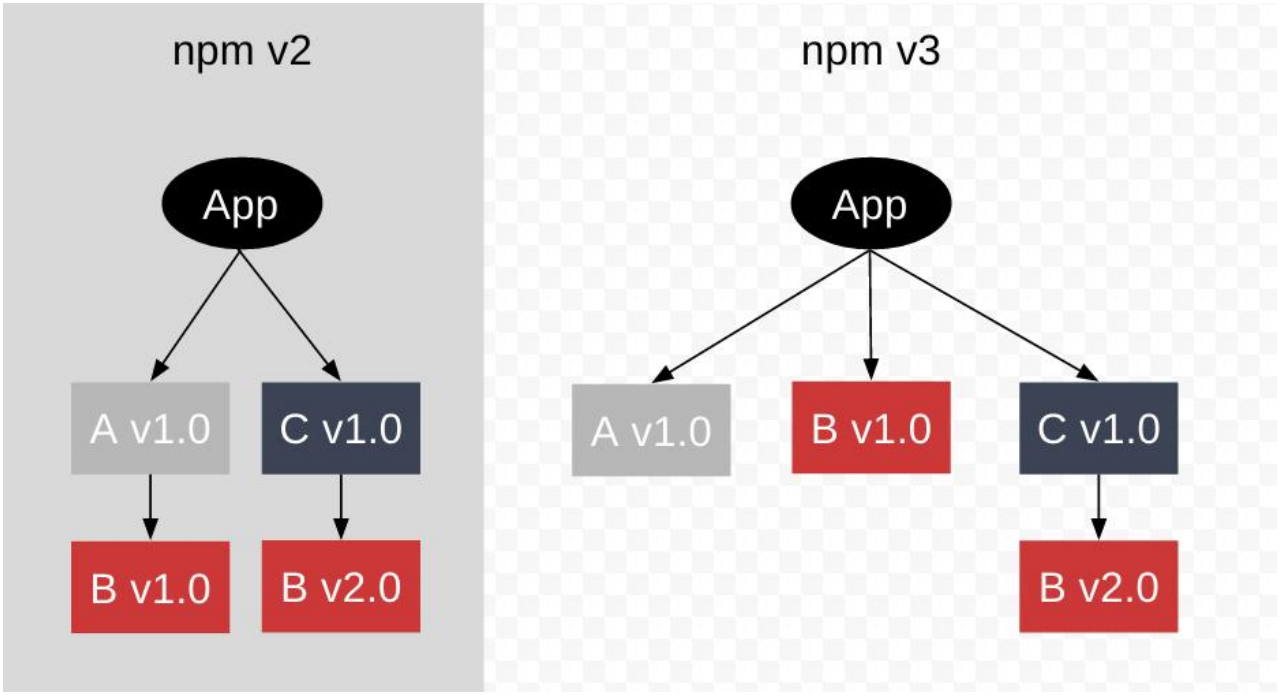
2016-01-01

npm3 的依赖管理方案

npm3 于2015年6月发布，它与 npm2 很大的一点不同是**依赖管理方案**的升级。

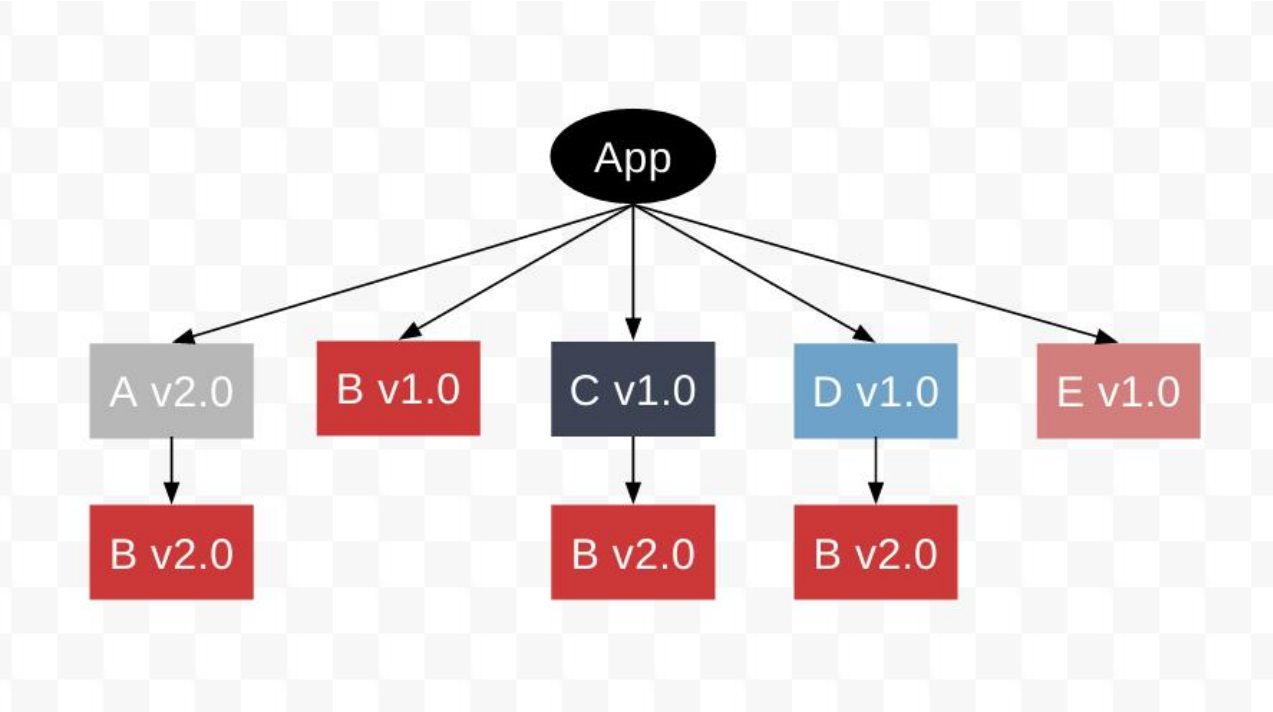
为了管理同一个模块的不同版本，npm2 采用严格树形嵌套的形式组织依赖模块的目录，而 npm3 则尽量*扁平化*，将依赖模块提升至顶层目录：



对于 B 模块的两个版本，v1 版本会被放置于顶层，而 v2 版本则因为冲突关系仍放置于 C 模块下面。

这样的布置有什么好处？共用。加入再有一个模块 D 依赖与 Bv1，则不必再安装，直接使用顶层的 Bv1 即可。

那如果 A 和 D 模块都依赖 Bv2 呢？npm3 不会将 Bv2 移至顶层，而是将 Bv2 仍挂在 C 和 D 下面：



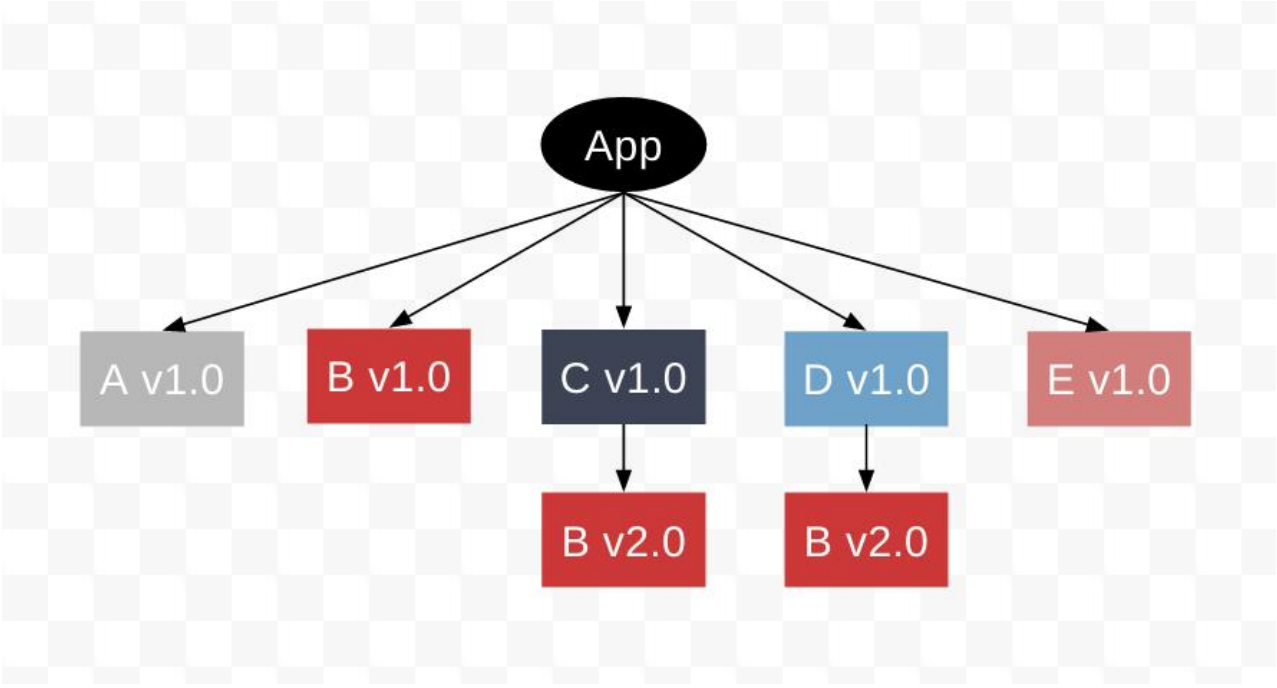
似乎 npm3 并没有那么优化到最好，Bv2 模块没有被复用。要想实现这种最优的组织，需要手动执行命令：

```
$npm dedupe
```

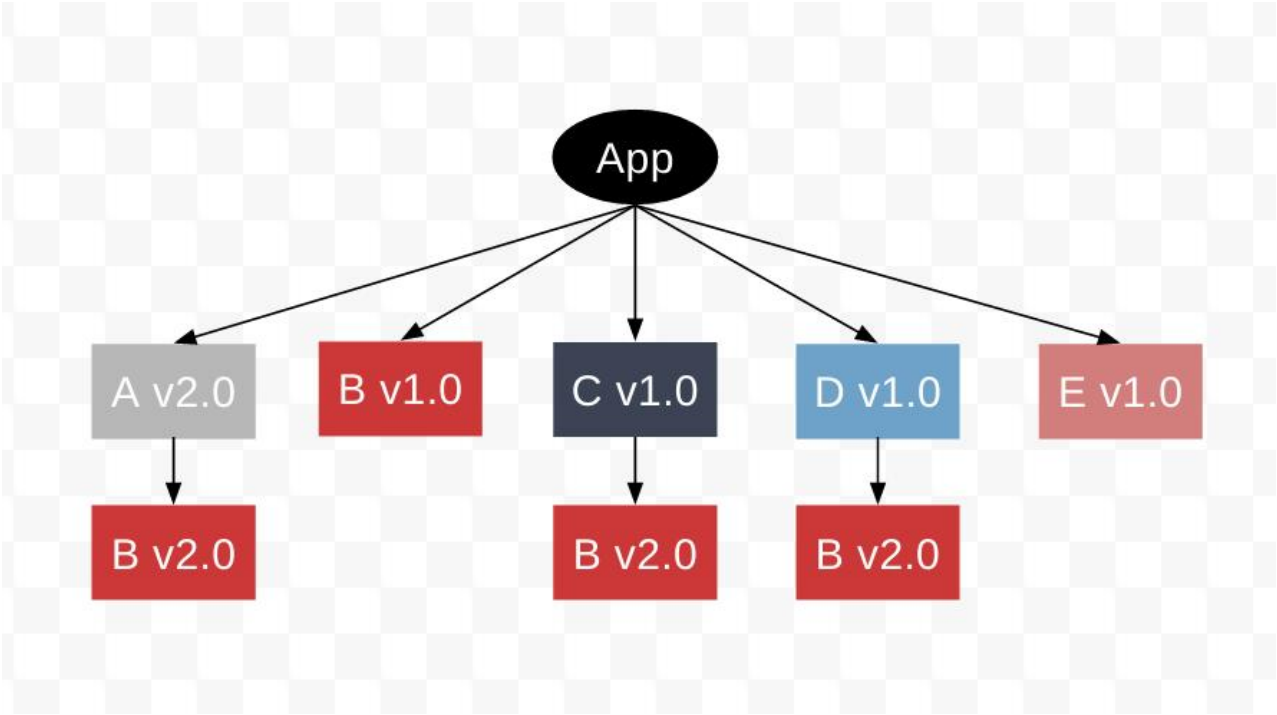
那么，npm3 如何决定 B 模块的哪一个版本在顶层呢？答案是按照自然顺序的先后，最先安装的版本会放置于顶层。

理论上，自然顺序在任何情况下都是一定的，因此依赖模块的最终组织形式也是一定的。但这个前提是安装依赖之前 node_modules 是空的。一旦 node_modules 内已经有依赖模块，最终的组织形式就会受到影响。

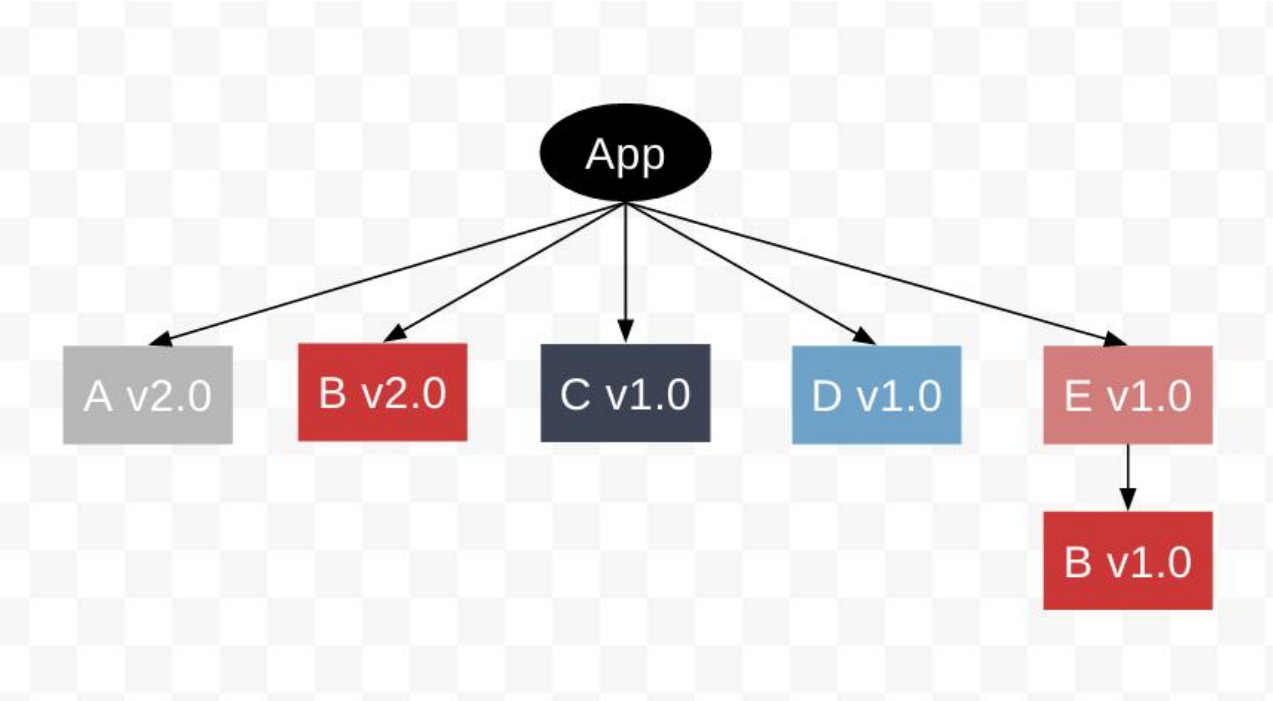
假如我们现在有这样的一个应用：



修改 A 模块，使之依赖 Bv2：



现在，发布应用，在另一台机器上重新部署应用：



可见这两种环境下，依赖的目录组织形式不是一致的。

因此，npm3 的这种新的依赖管理方式可能造成依赖模块的目录不一致，除非所有依赖模块都删除并重新安装。但是，这种不一致理论上是无害的，每个模块都能找到所有符合要求的依赖。

- https://docs.npmjs.com/how-npm-works/npm3
- https://docs.npmjs.com/how-npm-works/npm3-dupe
- https://docs.npmjs.com/how-npm-works/npm3-nondet

#dependencies #npm