How npm3 Works

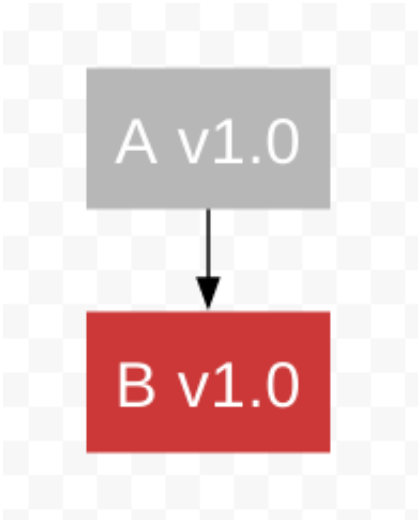
npm3 resolves dependencies differently than npm2.

While npm2 installs all dependencies in a nested way, npm3 tries to mitigate the deep trees and redundancy that such nesting causes. npm3 attempts this by installing some secondary dependencies (dependencies of dependencies) in a flat way, in the same directory as the primary dependency that requires it.

The key major differences are:

- position in the directory structure no longer predicts the type (primary, secondary, etc) a dependency is
- dependency resolution depends on *install order*, or the order in which things are installed will change the `node_modules` directory tree structure

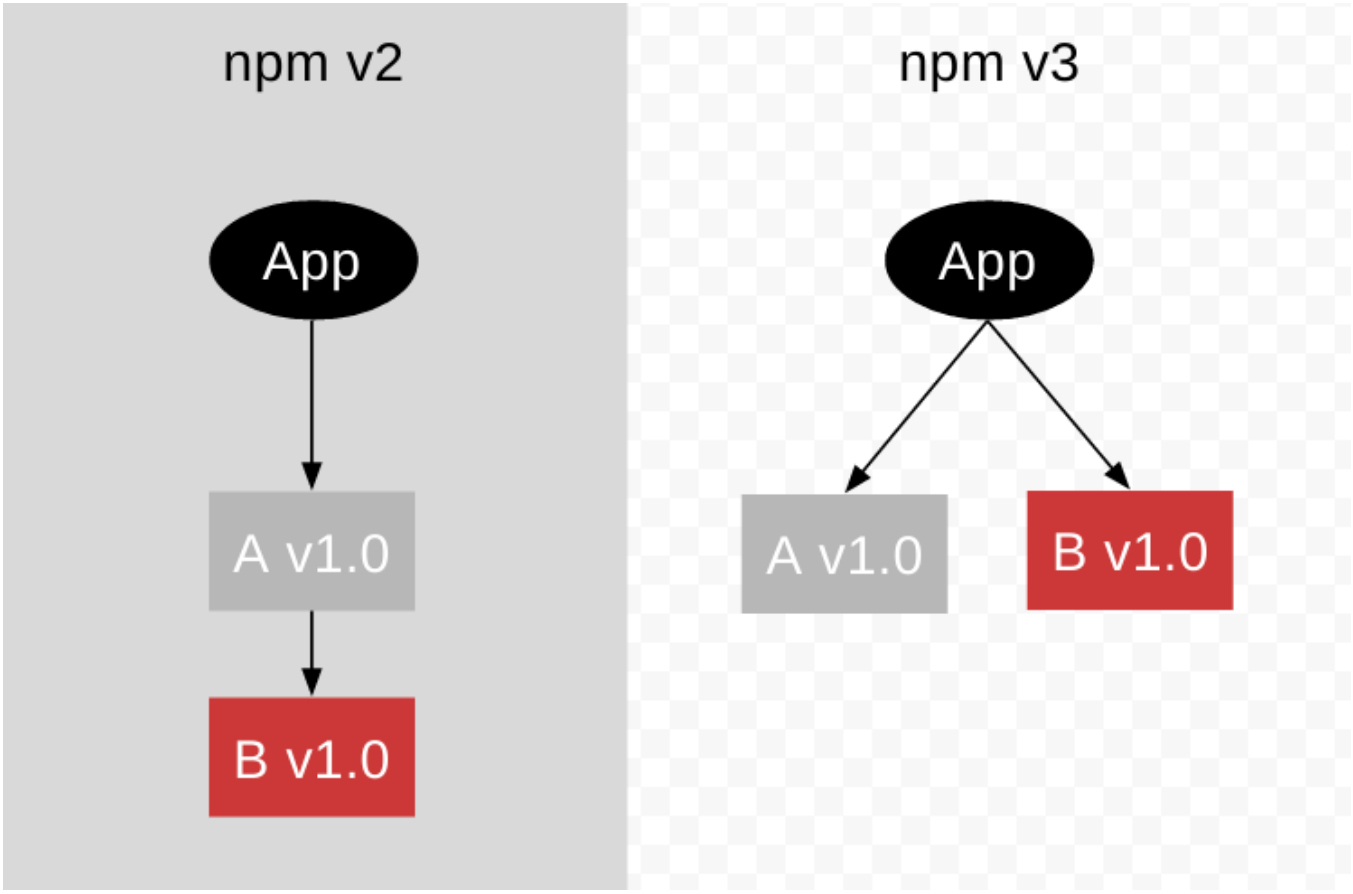
Imagine we have a module, A. A requires B.



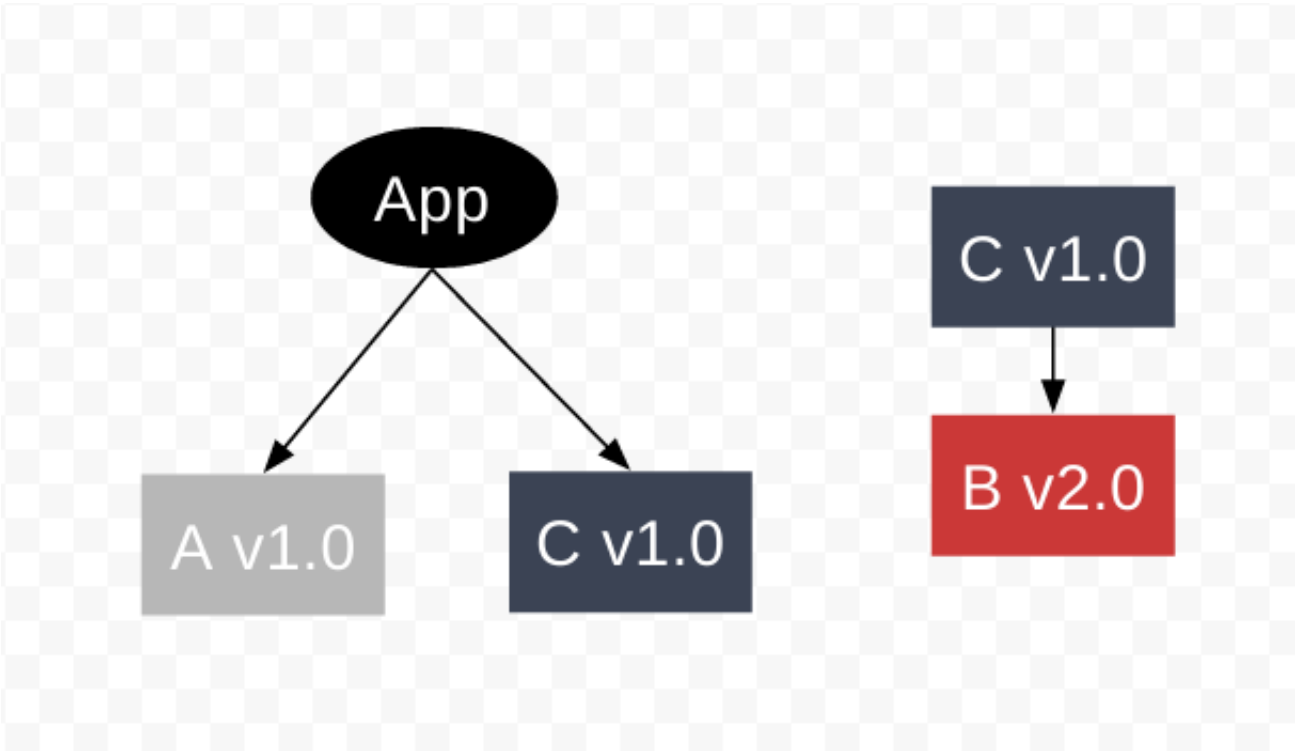
Now, let's create an application that requires module A.

On `npm install`, npm v3 will install both module A and its dependency, module B, inside the `/node_modules` directory, flat.

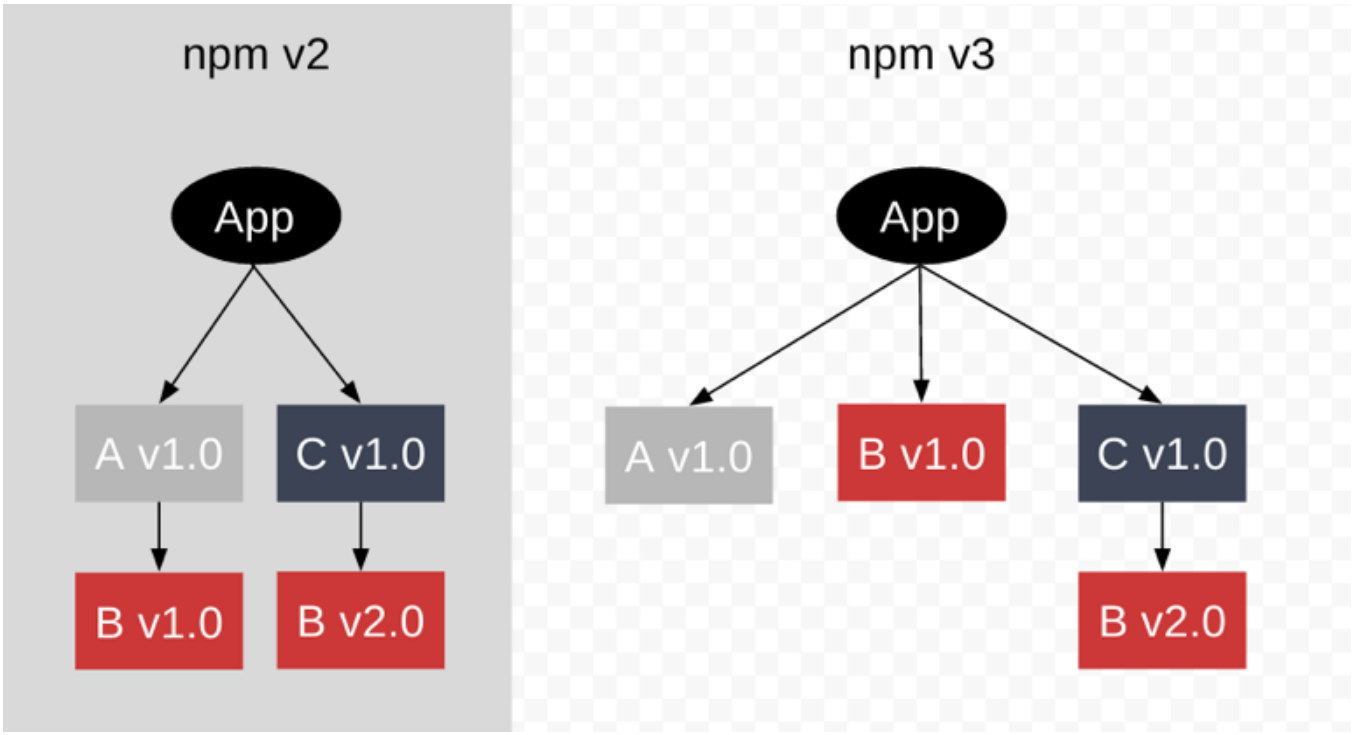
In npm v2 this would have happened in a nested way.



Now, let's say we want to require another module, C. C requires B, but at another version than A.



However, since B v1.0 is already a top-level dep, we cannot install B v2.0 as a top level dependency. npm v3 handles this by defaulting to npm v2 behavior and nesting the new, different, module B version dependency under the module that requires it -- in this case, module C.



In the terminal, this looks like this:

```
(jessie)ag_dubs@localhost:~/Projects/npm-sandbox/npm3/example1$ npm install mod-a --save
example1@1.0.0 /home/ag_dubs/Projects/npm-sandbox/npm3/example1
└─┬ mod-a@1.0.0
  └── mod-b@1.0.0

npm WARN example1@1.0.0 No description
npm WARN example1@1.0.0 No repository field.
(jessie)ag_dubs@localhost:~/Projects/npm-sandbox/npm3/example1$ npm install mod-c --save
example1@1.0.0 /home/ag_dubs/Projects/npm-sandbox/npm3/example1
└─┬ mod-c@1.0.0
  └── mod-b@2.0.0

npm WARN example1@1.0.0 No description
npm WARN example1@1.0.0 No repository field.
(jessie)ag_dubs@localhost:~/Projects/npm-sandbox/npm3/example1$ tree -d node_modules/
node_modules/
├── mod-a
├── mod-b
└── mod-c
    └── node_modules
        └── mod-b

5 directories
(jessie)ag_dubs@localhost:~/Projects/npm-sandbox/npm3/example1$ npm ls
example1@1.0.0 /home/ag_dubs/Projects/npm-sandbox/npm3/example1
├─┬ mod-a@1.0.0
│ └── mod-b@1.0.0
└─┬ mod-c@1.0.0
  └── mod-b@2.0.0
```

You can list the dependencies and still see their relationships using `npm ls`:

```
(jessie)ag_dubs@localhost:~/Projects/npm-sandbox/npm3/example1$ npm ls
example1@1.0.0 /home/ag_dubs/Projects/npm-sandbox/npm3/example1
├─┬ mod-a@1.0.0
│ └── mod-b@1.0.0
└─┬ mod-c@1.0.0
  └── mod-b@2.0.0
```

If you want to just see your primary dependencies, you can use:

```
npm ls --depth=0
```

```
(jessie)ag_dubs@localhost:~/Projects/npm-sandbox/npm3/example1$ npm ls --depth=0
example1@1.0.0 /home/ag_dubs/Projects/npm-sandbox/npm3/example1
├── mod-a@1.0.0
└── mod-c@1.0.0
```