从 npm 迁移到 yarn

Migrating from npm should be a fairly easy process for most users. Yarn can consume the same `package.json` format as npm, and can install any package from the npm registry.

If you want to try Yarn out on your existing npm project, just try running:

```
yarn
```

This will lay out your `node_modules` folder using Yarn's resolution algorithm that is compatible with the node.js module resolution algorithm.

If you get an error, please check for an existing issue or report it to the Yarn issue tracker.

When you run either `yarn` or `yarn add <package>`, Yarn will generate a `yarn.lock` file within the root directory of your package. You don't need to read or understand this file - just check it into source control. When other people start using Yarn instead of `npm`, the `yarn.lock` file will ensure that they get precisely the same dependencies as you have.

In most cases, running `yarn` or `yarn add` for the first time will just work. In some cases, the information in a `package.json` file is not explicit enough to eliminate dependencies, and the deterministic way that Yarn chooses dependencies will run into dependency conflicts. This is especially likely to happen in larger projects where sometimes `npm install` does not work and developers are frequently removing `node_modules` and rebuilding from scratch. If this happens, try using `npm` to make the versions of dependencies more explicit, before converting to Yarn.

As of Yarn 1.7.0, you can import your package-lock.json state, generated by `npm` to Yarn, by using `yarn import`.

Other developers on the project can keep using `npm`, so you don't need to get everyone on your project to convert at the same time. The developers using `yarn` will all get exactly the same configuration as each other, and the developers using `npm` may get slightly different configurations, which is the intended behavior of `npm`.

Later, if you decide that Yarn is not for you, you can just go back to using `npm` without making any particular changes. You can delete your old `yarn.lock` file if nobody on the project is using Yarn any more but it's not necessary.

If you are using an `npm-shrinkwrap.json` file right now, be aware that you may end up with a different set of dependencies. Yarn does not support npm shrinkwrap files as they don't have enough information in them to power Yarn's more deterministic algorithm. If you are using a shrinkwrap file it may be easier to convert everyone working on the project to use Yarn at the same time. Simply remove your existing `npm-shrinkwrap.json` file and check in the newly created `yarn.lock` file.

CLI commands comparison

npm (v5)	Yarn
`npm install`	`yarn install`
(N/A)	`yarn install --flat`
(N/A)	`yarn install --har`
`npm install --no-package-lock`	`yarn install --no-lockfile`
(N/A)	`yarn install --pure-lockfile`
`npm install [package]`	`yarn add [package]`
`npm install [package] --save-dev`	`yarn add [package] --dev`
(N/A)	`yarn add [package] --peer`
`npm install [package] --save-optional`	`yarn add [package] --optional`
`npm install [package] --save-exact`	`yarn add [package] --exact`
(N/A)	`yarn add [package] --tilde`
`npm install [package] --global`	`yarn global add [package]`

npm (v5)	Yarn
`npm update --global`	`yarn global upgrade`
`npm rebuild`	`yarn add --force`
`npm uninstall [package]`	`yarn remove [package]`
`npm cache clean`	`yarn cache clean [package]`
`rm -rf node_modules && npm install`	`yarn upgrade`
`npm version major`	`yarn version --major`
`npm version minor`	`yarn version --minor`
`npm version patch`	`yarn version --patch`

从 npm 迁移到 yarn

Yarn
基于 BSD 协议发布
行为准则

编辑本页