Atmosphere vs. npm

Building an application completely from scratch is a tall order. This is one of the main reasons you might consider using Meteor in the first place - you can focus on writing the code that is specific to your app, instead of reinventing wheels like user login and data synchronization. To streamline your workflow even further, it makes sense to use community packages from npm and Atmosphere. Many of these packages are recommended in the guide, and you can find more in the online directories.

With the release of version 1.3, Meteor has full support for npm. In the future, there will be a time when all packages will be migrated to npm, but currently there are benefits to both systems.

When to use Atmosphere packages

Atmosphere packages are packages written specifically for Meteor and have several advantages over npm when used with Meteor. In particular, Atmosphere packages can:

- Depend on core Meteor packages, such as `ddp` and `blaze`
- Explicitly include non-javascript files including CSS, Less, Sass, Stylus and static assets
- Take advantage of Meteor's build system to be automatically transpiled from languages like CoffeeScript
- Have a well defined way to ship different code for client and server, enabling different behavior in each context
- Get direct access to Meteor's package namespacing and package global exports without having to explicitly use ES2015 `import`
- Enforce exact version dependencies between packages using Meteor's constraint resolver
- Include build plugins for Meteor's build system
- Include pre-built binary code for different server architectures, such as Linux or Windows

If your package depends on an Atmosphere package (which, in Meteor 1.3, includes the Meteor core packages), or needs to take advantage of Meteor's build system, writing an Atmosphere package might be the best option for now.

When to use npm packages

npm is a repository of general JavaScript packages. These packages were originally intended solely for the Node.js server-side environment, but as the JavaScript ecosystem matured, solutions arose to enable the use of npm packages in other environments such as the browser. Today, npm is used for all types of JavaScript packages.

If you want to distribute and reuse code that you've written for a Meteor application, then you should consider publishing that code on npm if it's general enough to be consumed by a wider JavaScript audience. It's possible to use npm packages in Meteor applications, and possible to use npm packages within Atmosphere packages, so even if your main audience is Meteor developers, npm might be the best choice.

> Meteor comes with npm bundled so that you can type `meteor npm` without worrying about installing it yourself. If you like, you can also use a globally installed npm to manage your packages.