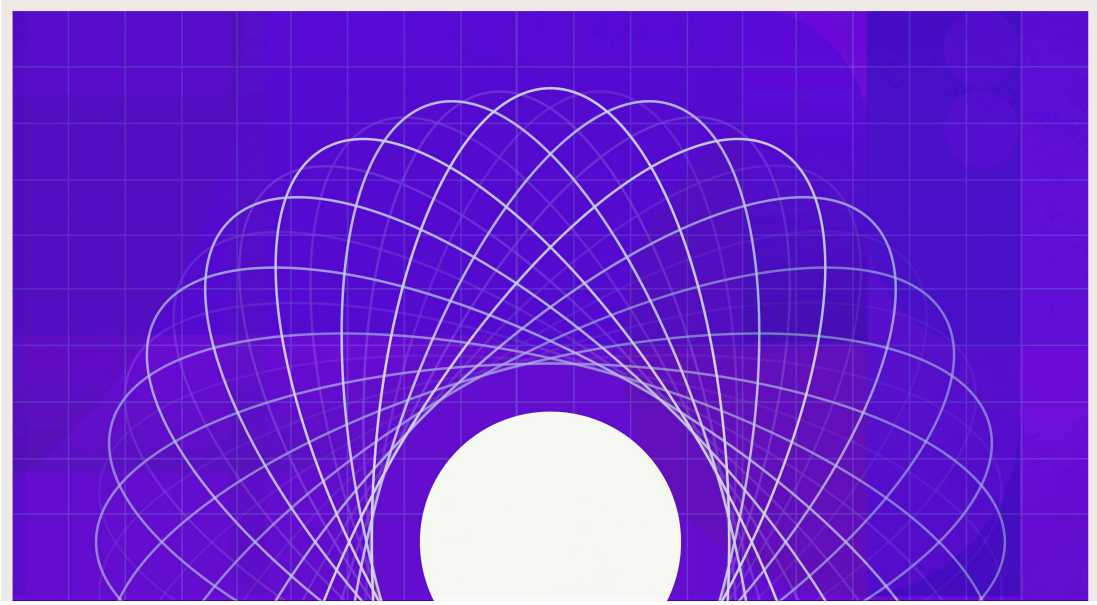
Nylas

Back to Nylas Blogs

The Nylas Engineering Blog



Our new build process for N1: Improving Electron every day

How we decreased N1 download size & improved performance

By: The Nylas Team

December 11, 2016

We started working on N1 almost two years ago. At the time, Electron didn't yet exist and so we began the project by forking the Atom code editor. When we did that, we inherited a lot of the manual tooling that GitHub had built to test Atom and ship new releases. For the last two years, we've still been using those scripts—basically in their original form—with a fair amount of success.

But in the last two years, we've seen the Electron community move forward and solve a lot of problems in better ways, with less manual tooling and less cruft.

That's why we recently adopted two new projects in N1: electron-compile and electron-packager.

Electron-compile

Electron-compile is a new open source project from the Slack team that makes it easy to build Electron apps that support languages other than stock JavaScript—things like TypeScript, CoffeeScript, LESS, Jade, and Babel. We already supported those languages in N1 with Atom, but we also had almost 3,000 lines of our own code on top. Several members of the Nylas team are also active contributors to the project.

Using electron-compile allowed us to rip out a lot of the homegrown tooling that we were using. Instead, we're now leveraging a project that is growing really quickly and being maintained by the entire Electron community. As we grow, this approach will make it easier for us to manage and maintain the build process.

Electron-packager

We're now also using electron-packager for all of our build and installation generation. This makes it really easy for us to package N1 for distribution on Mac and Windows. Switching to electron-packager is a very similar story—our build process was several thousand lines of code that was almost two years old and based on Atom. But folks on the team were nervous to change things because a lot of it was brittle and confusing.

Using this new electron-packager project has made it a lot easier for us to clean up the release processes and switch to a tool with great documentation. Now if an engineer wants to go make changes to the build process or make changes to the installation process, they can get up to speed with tutorials, guides, and full docs for these tools instead of digging through our previous homegrown build scripts.

It's great to see these things coming together in the Electron community and we're excited to contribute back our own modifications and tools as we continue developing N1.

Better language support and smaller downloads

Thanks to our adoption of these two Electron projects, N1 supports a wider set of languages than it used to. For example, now you can write HTML into a plugin using Jade transformations. This and much more is supported out-of-the-box by the electron-compile project.

This new build process has also optimized several parts of our application resulting in a smaller final build size. When you download N1 today, it's about 10 megabytes smaller. (That might not sound like a lot, but it makes a big difference when you have a slower internet connection!)

Driving the entire community forward

As a small team trying to quickly ship a product cross-platform for Mac, Windows, and Linux, ease of maintenance is a pretty big deal for us. Using electron-compile and electron-packager means that we can spend more time shipping features and doing cool stuff. By sharing some of the packaging work with other companies like Slack, we can help eliminate duplicate work for everyone in the Electron community. The rising tide floats all boats.

Going forward

If you're considering building an Electron app, we recommend leveraging electron-compile and electron-packager to avoid home-growing a solution on their own. If you do, your app will be faster and support more technologies—and it'll be easier for you to maintain.

Have questions or comments for new ways to use Electron? Let us know on Twitter here.