Boilerplates and CLIs

Electron development is unopinionated - there is no "one true way" to develop, build, package, or release an Electron application. Additional features for Electron, both for build- and run-time, can usually be found on npm in individual packages, allowing developers to build both the app and build pipeline they need.

That level of modularity and extendability ensures that all developers working with Electron, both big and small in team-size, are never restricted in what they can or cannot do at any time during their development lifecycle. However, for many developers, one of the community-driven boilerplates or command line tools might make it dramatically easier to compile, package, and release an app.

Boilerplate vs CLI

A boilerplate is only a starting point - a canvas, so to speak - from which you build your application. They usually come in the form of a repository you can clone and customize to your heart's content.

A command line tool on the other hand continues to support you throughout the development and release. They are more helpful and supportive but enforce guidelines on how your code should be structured and built. *Especially for beginners, using a command line tool is likely to be helpful.*

electron-forge

A "complete tool for building modern Electron applications". Electron Forge unifies the existing (and well maintained) build tools for Electron development into a cohesive package so that anyone can jump right in to Electron development.

Forge comes with a ready-to-use template using Webpack as a bundler. It includes an example typescript configuration and provides two configuration files to enable easy customization. It uses the same core modules used by the greater Electron community (like `electron-packager`) – changes made by Electron maintainers (like Slack) benefit Forge's users, too.

You can find more information and documentation on electronforge.io.

electron-builder

A "complete solution to package and build a ready-for-distribution Electron app" that focuses on an integrated experience. `electron-builder` adds one single dependency focused on simplicity and manages all further requirements internally.

`electron-builder` replaces features and modules used by the Electron maintainers (such as the auto-updater) with custom ones. They are generally tighter integrated but will have less in common with popular Electron apps like Atom, Visual Studio Code, or Slack.

You can find more information and documentation in the repository.

electron-react-boilerplate

If you don't want any tools but only a solid boilerplate to build from, CT Lin's `electron-react-boilerplate` might be worth a look. It's quite popular in the community and uses `electron-builder` internally.

Other Tools and Boilerplates

The "Awesome Electron" list contains more tools and boilerplates to choose from. If you find the length of the list intimidating, don't forget that adding tools as you go along is a valid approach, too.