Developing Electron

These guides are intended for people working on the Electron project itself. For guides on Electron app development, see /docs/README.md.

Table of Contents

Getting Started

In order to contribute to Electron, the first thing you'll want to do is get the code.

Electron's `build-tools` automate much of the setup for compiling Electron from source with different configurations and build targets.

If you would prefer to build Electron manually, see the build instructions.

Once you've checked out and built the code, you may want to take a look around the source tree to get a better idea of what each directory is responsible for. The source code directory structure gives a good overview of the purpose of each directory.

Opening Issues on Electron

For any issue, there are generally three ways an individual can contribute:

1. By opening the issue for discussion
   - If you believe that you have found a new bug in Electron, you should report it by creating a new issue in the `electron/electron` issue tracker.
2. By helping to triage the issue
   - You can do this either by providing assistive details (a reproducible test case that demonstrates a bug) or by providing suggestions to address the issue.
3. By helping to resolve the issue
   - This can be done by demonstrating that the issue is not a bug or is fixed; but more often, by opening a pull request that changes the source in `electron/electron` in a concrete and reviewable manner.

See issues for more information.

Making a Pull Request to Electron

Most pull requests opened against the `electron/electron` repository include changes to either the C/C++ code in the `shell/` folder, the TypeScript code in the `lib/` folder, the documentation in `docs/`, or tests in the `spec/` and `spec-main/` folders.

See pull requests for more information.

If you want to add a new API module to Electron, you'll want to look in creating API.

Governance

Electron has a fully-fledged governance system that oversees activity in Electron and whose working groups are responsible for areas like APIs, releases, and upgrades to Electron's dependencies including Chromium and Node.js. Depending on how frequently and to what end you want to contribute, you may want to consider joining a working group.

Details about each group and their reponsibilities can be found in the governance repo.

Patches in Electron

Electron is built on two major upstream projects: Chromium and Node.js. Each of these projects has several of their own dependencies, too. We try our best to use these dependencies exactly as they are but sometimes we can't achieve our goals without patching those upstream dependencies to fit our use cases.

As such, we maintain a collection of patches as part of our source tree. The process for adding or altering one of these patches to Electron's source tree via a pull request can be found in patches.

Debugging

There are many different approaches to debugging issues and bugs in Electron, many of which are platform specific.

For an overview of information related to debugging Electron itself (and not an app *built with Electron*), see debugging.