Application Distribution

Overview

To distribute your app with Electron, you need to package and rebrand it. To do this, you can either use specialized tooling or manual approaches.

With tooling

You can use the following tools to distribute your application:

- electron-forge
- electron-builder
- electron-packager

These tools will take care of all the steps you need to take to end up with a distributable Electron application, such as bundling your application, rebranding the executable, and setting the right icons.

You can check the example of how to package your app with `electron-forge` in the Quick Start guide.

Manual distribution

With prebuilt binaries

To distribute your app manually, you need to download Electron's prebuilt binaries. Next, the folder containing your app should be named `app` and placed in Electron's resources directory as shown in the following examples.

> *NOTE: the location of Electron's prebuilt binaries is indicated with `electron/` in the examples below.*

On macOS:

```
electron/Electron.app/Contents/Resources/app/
├── package.json
├── main.js
└── index.html
```

On Windows and Linux:

```
electron/resources/app
├── package.json
├── main.js
└── index.html
```

Then execute `Electron.app` on macOS, `electron` on Linux, or `electron.exe` on Windows, and Electron will start as your app. The `electron` directory will then be your distribution to deliver to users.

With an app source code archive

Instead of shipping your app by copying all of its source files, you can package your app into an asar archive to improve the performance of reading files on platforms like Windows, if you are not already using a bundler such as Parcel or Webpack.

To use an `asar` archive to replace the `app` folder, you need to rename the archive to `app.asar`, and put it under Electron's resources directory like below, and Electron will then try to read the archive and start from it.

On macOS:

```
electron/Electron.app/Contents/Resources/
└── app.asar
```

On Windows and Linux:

```
electron/resources/
└── app.asar
```

You can find more details on how to use `asar` in the `electron/asar` repository.

Rebranding with downloaded binaries

After bundling your app into Electron, you will want to rebrand Electron before distributing it to users.

macOS

You can rename `Electron.app` to any name you want, and you also have to rename the `CFBundleDisplayName`, `CFBundleIdentifier` and `CFBundleName` fields in the following files:

- `Electron.app/Contents/Info.plist`
- `Electron.app/Contents/Frameworks/Electron Helper.app/Contents/Info.plist`

You can also rename the helper app to avoid showing `Electron Helper` in the Activity Monitor, but make sure you have renamed the helper app's executable file's name.

The structure of a renamed app would be like:

```
MyApp.app/Contents
├── Info.plist
├── MacOS/
│   └── MyApp
└── Frameworks/
    └── MyApp Helper.app
        ├── Info.plist
        └── MacOS/
            └── MyApp Helper
```

Windows

You can rename `electron.exe` to any name you like, and edit its icon and other information with tools like rcedit.

Linux

You can rename the `electron` executable to any name you like.

Rebranding by rebuilding Electron from source

It is also possible to rebrand Electron by changing the product name and building it from source. To do this you need to set the build argument corresponding to the product name (`electron_product_name = "YourProductName"`) in the `args.gn` file and rebuild.