You decided to contribute to this project? Great, thanks a lot for pushing it.

This project adheres to the [Contributor Covenant](http://contributor-covenant.org) code of conduct. By participating, you are expected to uphold this code. Please file issue to report unacceptable behavior.

## Pull Requests

To check that your contributions match the project coding style make sure `yarn test` passes.

[yarn](https://yarnpkg.com) is required because NPM is not reliable.

To build project: `yarn && yarn compile`

If you get strange compilation errors, try to remove all `node\_modules` in the project (especially under `packages/\*`).

### Git Commit Guidelines

We use [semantic-release](https://github.com/semantic-release/semantic-release), so we have very precise rules over how our git [commit messages can be formatted](https://gist.github.com/develar/273e2eb938792cf5f86451fbac2bcd51).

## Documentation

To avoid google indexing, documentation files located in the branch `docs`. To clone:

```shell script

git clone --single-branch -b docs git@github.com:electron-userland/electron-builder.git docs

```

Documentation files located in the `/docs`.

`/docs` is deployed to Netlify when `next` release is marked as `latest` and available for all users.

`pip3 install mkdocs-material mkdocs markdown-include pymdown-extensions pygments --upgrade`

You'll want to copy the `mkdocs.yml` file from the master branch and then: `mkdocs build`

## Debug Tests

Only IntelliJ Platform IDEs ([IntelliJ IDEA](https://confluence.jetbrains.com/display/IDEADEV/IDEA+2017.1+EAP), [WebStorm](https://confluence.jetbrains.com/display/WI/WebStorm+EAP)) support debug.

If you use IntelliJ IDEA or WebStorm — [ij-rc-producer](https://github.com/develar/ij-rc-producer) is used and you can run tests from an editor (just click on `Run` green gutter icon).

Or you can create Node.js run configuration manually:

\* Ensure that `Before launch` contains `Compile TypeScript`.

\* Set `Node interpreter` to NodeJS 8. NodeJS 8 is required to debug.

\* Set `Application Parameters` to `-t "test name" relative-test-file-name` if you want to debug particular test. E.g.

```

-t "extraResources - one-package" globTest.js

```

\* Set `Environment Variables`:

\* Optionally, `TEST\_APP\_TMP\_DIR` to some directory (e.g. `/tmp/electron-builder-test`) to inspect output if test uses temporary directory (only if `--match` is used). Specified directory will be used instead of random temporary directory and \*cleared\* on each run.

### Run Test using CLI

```sh

TEST\_APP\_TMP\_DIR=/tmp/electron-builder-test ./node\_modules/.bin/jest --env jest-environment-node-debug -t 'assisted' '/oneClickInstallerTest\.\w+$'

```

where `TEST\_APP\_TMP\_DIR` is specified to easily inspect and use test build, `assisted` is the test name and `/oneClickInstallerTest\.\w+$` is the path to test file.

Do not forget to execute `yarn compile` before run.

## Issues

When filing an issue please make sure, that you give all information needed.

This includes:

- description of what you're trying to do

- package.json

- log of the terminal output

- node version

- npm version

- on which system do you want to create installers (macOS, Linux or Windows).