# Testing

We aim to keep the code coverage of Electron high. We ask that all pull request not only pass all existing tests, but ideally also add new tests to cover changed code and new scenarios. Ensuring that we capture as many code paths and use cases of Electron as possible ensures that we all ship apps with fewer bugs.

This repository comes with linting rules for both JavaScript and C++ – as well as unit and integration tests. To learn more about Electron's coding style, please see the coding-style document.

## Linting

To ensure that your changes are in compliance with the Electron coding style, run `npm run lint`, which will run a variety of linting checks against your changes depending on which areas of the code they touch.

Many of these checks are included as precommit hooks, so it's likely you error would be caught at commit time.

## Unit Tests

If you are not using build-tools, ensure that the name you have configured for your local build of Electron is one of `Testing`, `Release`, `Default`, or you have set `process.env.ELECTRON_OUT_DIR`. Without these set, Electron will fail to perform some pre-testing steps.

To run all unit tests, run `npm run test`. The unit tests are an Electron app (surprise!) that can be found in the `spec` folder. Note that it has its own `package.json` and that its dependencies are therefore not defined in the top-level `package.json`.

To run only tests in a specific process, run `npm run test --runners=PROCESS` where `PROCESS` is one of `main` or `remote`.

To run only specific tests matching a pattern, run `npm run test -- -g=PATTERN`, replacing the `PATTERN` with a regex that matches the tests you would like to run. As an example: If you want to run only IPC tests, you would run `npm run test -- -g ipc`.

## Node.js Smoke Tests

If you've made changes that might affect the way Node.js is embedded into Electron, we have a test runner that runs all of the tests from Node.js, using Electron's custom fork of Node.js.

To run all of the Node.js tests:

```
$ node script/node-spec-runner.js
```

To run a single Node.js test:

```
$ node script/node-spec-runner.js parallel/test-crypto-keygen
```

where the argument passed to the runner is the path to the test in the Node.js source tree.

### Testing on Windows 10 devices

**Extra steps to run the unit test:**

1. Visual Studio 2019 must be installed.
2. Node headers have to be compiled for your configuration.

   ```
   ninja -C out\Testing third_party\electron_node:headers
   ```

3. The electron.lib has to be copied as node.lib.

   ```
   cd out\Testing
   mkdir gen\node_headers\Release
   copy electron.lib gen\node_headers\Release\node.lib
   ```

**Missing fonts** Some Windows 10 devices do not ship with the Meiryo font installed, which may cause a font fallback test to fail. To install Meiryo:

1. Push the Windows key and search for *Manage optional features*.
2. Click *Add a feature*.
3. Select *Japanese Supplemental Fonts* and click *Install*.

**Pixel measurements** Some tests which rely on precise pixel measurements may not work correctly on devices with Hi-DPI screen settings due to floating point precision errors. To run these tests correctly, make sure the device is set to 100% scaling.

To configure display scaling:

1. Push the Windows key and search for *Display settings*.
2. Under *Scale and layout*, make sure that the device is set to 100%.