Contributing to Atom

:+1::tada: First off, thanks for taking the time to contribute! :tada::+1:

The following is a set of guidelines for contributing to Atom and its packages, which are hosted in the Atom Organization on GitHub. These are mostly guidelines, not rules. Use your best judgment, and feel free to propose changes to this document in a pull request.

Table Of Contents

Code of Conduct

This project and everyone participating in it is governed by the Atom Code of Conduct. By participating, you are expected to uphold this code. Please report unacceptable behavior to atom@github.com.

I don't want to read this whole thing I just have a question!!!

> ***Note:*** *Please don't file an issue to ask a question. You'll get faster results by using the resources below.*

We have an official message board with a detailed FAQ and where the community chimes in with helpful advice if you have questions.

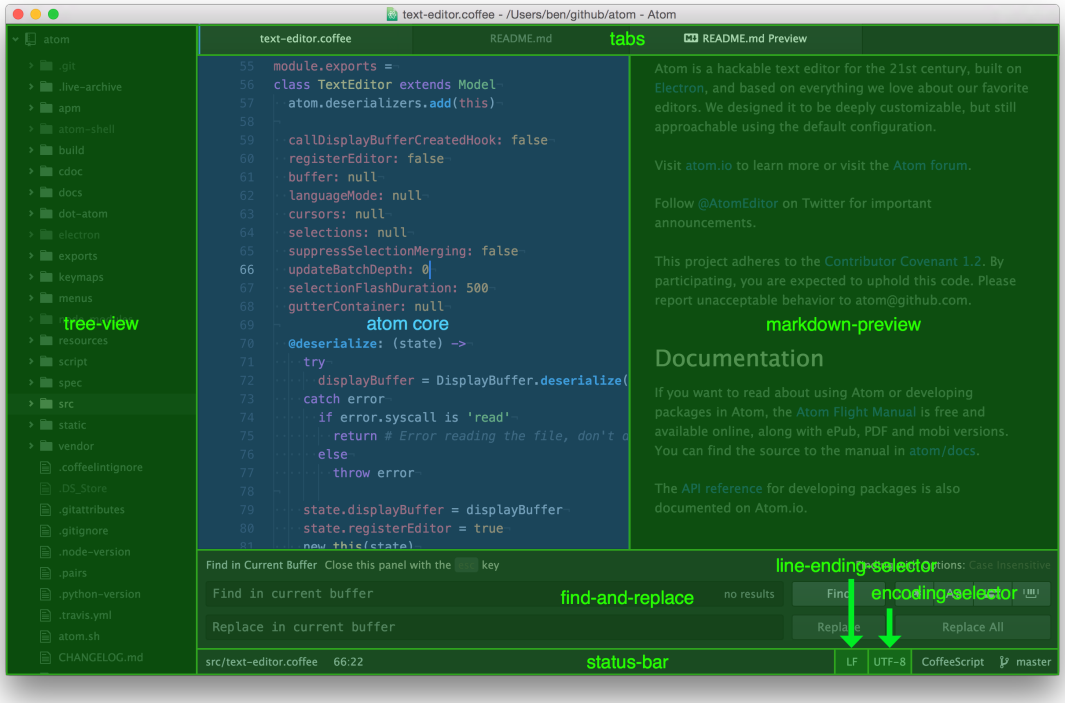
- Github Discussions, the official Atom message board
- Atom FAQ

What should I know before I get started?

Atom and Packages

Atom is a large open source project — it's made up of over 200 repositories. When you initially consider contributing to Atom, you might be unsure about which of those 200 repositories implements the functionality you want to change or report a bug for. This section should help you with that.

Atom is intentionally very modular. Nearly every non-editor UI element you interact with comes from a package, even fundamental things like tabs and the status-bar. These packages are packages in the same way that packages in the Atom package repository are packages, with one difference: they are bundled into the default distribution.



To get a sense for the packages that are bundled with Atom, you can go to `Settings` > `Packages` within Atom and take a look at the Core Packages section.

Here's a list of the big ones:

- atom/atom - Atom Core! The core editor component is responsible for basic text editing (e.g. cursors, selections, scrolling), text indentation, wrapping, and folding, text rendering, editor rendering, file system operations (e.g. saving), and installation and auto-updating. You should also use this repository for feedback related to the Atom API and for large, overarching design proposals.
- tree-view - file and directory listing on the left of the UI.
- fuzzy-finder - the quick file opener.
- find-and-replace - all search and replace functionality.
- tabs - the tabs for open editors at the top of the UI.
- status-bar - the status bar at the bottom of the UI.
- markdown-preview - the rendered markdown pane item.
- settings-view - the settings UI pane item.
- autocomplete-plus - autocompletions shown while typing. Some languages have additional packages for autocompletion functionality, such as autocomplete-html.
- git-diff - Git change indicators shown in the editor's gutter.

- language-javascript - all bundled languages are packages too, and each one has a separate package `language-[name]`. Use these for feedback on syntax highlighting issues that only appear for a specific language.
- one-dark-ui - the default UI styling for anything but the text editor. UI theme packages (i.e. packages with a `-ui` suffix) provide only styling and it's possible that a bundled package is responsible for a UI issue. There are other bundled UI themes, such as one-light-ui.
- one-dark-syntax - the default syntax highlighting styles applied for all languages. There are other bundled syntax themes, such as solarized-dark-syntax. You should use these packages for reporting issues that appear in many languages, but disappear if you change to another syntax theme.
- apm - the `apm` command line tool (Atom Package Manager). You should use this repository for any contributions related to the `apm` tool and for publishing packages.
- atom.io - the repository for feedback on the Atom.io website and the Atom.io package API used by apm.

There are many more, but this list should be a good starting point. For more information on how to work with Atom's official packages, see Contributing to Atom Packages.

Also, because Atom is so extensible, it's possible that a feature you've become accustomed to in Atom or an issue you're encountering isn't coming from a bundled package at all, but rather a community package you've installed. Each community package has its own repository too.

Package Conventions

There are a few conventions that have developed over time around packages:

- Packages that add one or more syntax highlighting grammars are named `language-[language-name]`
  - Language packages can add other things besides just a grammar. Many offer commonly-used snippets. Try not to add too much though.
- Theme packages are split into two categories: UI and Syntax themes
  - UI themes are named `[theme-name]-ui`
  - Syntax themes are named `[theme-name]-syntax`
  - Often themes that are designed to work together are given the same root name, for example: `one-dark-ui` and `one-dark-syntax`
  - UI themes style everything outside of the editor pane — all of the green areas in the packages image above
  - Syntax themes style just the items inside the editor pane, mostly syntax highlighting
- Packages that add autocomplete providers are named `autocomplete-[what-they-autocomplete]` — ex: autocomplete-css

Design Decisions

When we make a significant decision in how we maintain the project and what we can or cannot support, we will document it in the atom/design-decisions repository. If you have a question around how we do things, check to see if it is documented there. If it is *not* documented there, please open a new topic on Github Discussions, the official Atom message board and ask your question.

How Can I Contribute?

Reporting Bugs

This section guides you through submitting a bug report for Atom. Following these guidelines helps maintainers and the community understand your report :pencil:, reproduce the behavior :computer: :computer:, and find related reports :mag_right:.

Before creating bug reports, please check this list as you might find out that you don't need to create one. When you are creating a bug report, please include as many details as possible. Fill out the required template, the information it asks for helps us resolve issues faster.

> ***Note:*** *If you find a* ***Closed*** *issue that seems like it is the same thing that you're experiencing, open a new issue and include a link to the original issue in the body of your new one.*

Before Submitting A Bug Report

- **Check the debugging guide.** You might be able to find the cause of the problem and fix things yourself. Most importantly, check if you can reproduce the problem in the latest version of Atom, if the problem happens when you run Atom in safe mode, and if you can get the desired behavior by changing Atom's or packages' config settings.
- **Check the faq and the discussions** for a list of common questions and problems.
- **Determine which repository the problem should be reported in**.
- **Perform a cursory search** to see if the problem has already been reported. If it has **and the issue is still open**, add a comment to the existing issue instead of opening a new one.

How Do I Submit A (Good) Bug Report?

Bugs are tracked as GitHub issues. After you've determined which repository your bug is related to, create an issue on that repository and provide the following information by filling in the template.

Explain the problem and include additional details to help maintainers reproduce the problem:

- **Use a clear and descriptive title** for the issue to identify the problem.
- **Describe the exact steps which reproduce the problem** in as many details as possible. For example, start by explaining how you started Atom, e.g. which command exactly you used in the terminal, or how you started Atom otherwise. When listing steps, **don't just say what you did, but explain how you did it**. For example, if you moved the cursor to the end of a line, explain if you used the mouse, or a keyboard shortcut or an Atom command, and if so which one?
- **Provide specific examples to demonstrate the steps**. Include links to files or GitHub projects, or copy/pasteable snippets, which you use in those examples. If you're providing snippets in the issue, use Markdown code blocks.
- **Describe the behavior you observed after following the steps** and point out what exactly is the problem with that behavior.
- **Explain which behavior you expected to see instead and why.**
- **Include screenshots and animated GIFs** which show you following the described steps and clearly demonstrate the problem. If you use the keyboard while following the steps, **record the GIF with the Keybinding Resolver shown**. You can use this tool to record GIFs on macOS and Windows, and this tool or this tool on Linux.
- **If you're reporting that Atom crashed**, include a crash report with a stack trace from the operating system. On macOS, the crash report will be available in `Console.app` under "Diagnostic and usage information" > "User diagnostic reports". Include the crash report in the issue in a code block, a file attachment, or put it in a gist and provide link to that gist.
- **If the problem is related to performance or memory**, include a CPU profile capture with your report.
- **If Chrome's developer tools pane is shown without you triggering it**, that normally means that you have a syntax error in one of your themes or in your `styles.less`. Try running in Safe Mode and using a different theme or comment out the contents of your `styles.less` to see if that fixes the problem.
- **If the problem wasn't triggered by a specific action**, describe what you were doing before the problem happened and share more information using the guidelines below.

Provide more context by answering these questions:

- **Can you reproduce the problem in safe mode?**
- **Did the problem start happening recently** (e.g. after updating to a new version of Atom) or was this always a problem?
- If the problem started happening recently, **can you reproduce the problem in an older version of Atom?** What's the most recent version in which the problem doesn't happen? You can download older versions of Atom from the releases page.
- **Can you reliably reproduce the issue?** If not, provide details about how often the problem happens and under which conditions it normally happens.
- If the problem is related to working with files (e.g. opening and editing files), **does the problem happen for all files and projects or only some?** Does the problem happen only when working with local or remote files (e.g. on network drives), with files of a specific type (e.g. only JavaScript or Python files), with large files or files with very long lines, or with files in a specific encoding? Is there anything else special about the files you are using?

Include details about your configuration and environment:

- **Which version of Atom are you using?** You can get the exact version by running `atom -v` in your terminal, or by starting Atom and running the `Application: About` command from the Command Palette.
- **What's the name and version of the OS you're using**?
- **Are you running Atom in a virtual machine?** If so, which VM software are you using and which operating systems and versions are used for the host and the guest?
- **Which packages do you have installed?** You can get that list by running `apm list --installed`.
- **Are you using local configuration files** `config.cson`, `keymap.cson`, `snippets.cson`, `styles.less` and `init.coffee` to customize Atom? If so, provide the contents of those files, preferably in a code block or with a link to a gist.
- **Are you using Atom with multiple monitors?** If so, can you reproduce the problem when you use a single monitor?
- **Which keyboard layout are you using?** Are you using a US layout or some other layout?

Suggesting Enhancements

This section guides you through submitting an enhancement suggestion for Atom, including completely new features and minor improvements to existing functionality. Following these guidelines helps maintainers and the community understand your suggestion :pencil: and find related suggestions :mag_right:.

Before creating enhancement suggestions, please check this list as you might find out that you don't need to create one. When you are creating an enhancement suggestion, please include as many details as possible. Fill in the template, including the steps that you imagine you would take if the feature you're requesting existed.

Before Submitting An Enhancement Suggestion

- **Check the debugging guide** for tips — you might discover that the enhancement is already available. Most importantly, check if you're using the latest version of Atom and if you can get the desired behavior by changing Atom's or packages' config settings.
- **Check if there's already a package which provides that enhancement.**
- **Determine which repository the enhancement should be suggested in.**
- **Perform a cursory search** to see if the enhancement has already been suggested. If it has, add a comment to the existing issue instead of opening a new one.

How Do I Submit A (Good) Enhancement Suggestion?

Enhancement suggestions are tracked as GitHub issues. After you've determined which repository your enhancement suggestion is related to, create an issue on that repository and provide the following information:

- **Use a clear and descriptive title** for the issue to identify the suggestion.
- **Provide a step-by-step description of the suggested enhancement** in as many details as possible.
- **Provide specific examples to demonstrate the steps**. Include copy/pasteable snippets which you use in those examples, as Markdown code blocks.
- **Describe the current behavior** and **explain which behavior you expected to see instead** and why.
- **Include screenshots and animated GIFs** which help you demonstrate the steps or point out the part of Atom which the suggestion is related to. You can use this tool to record GIFs on macOS and Windows, and this tool or this tool on Linux.
- **Explain why this enhancement would be useful** to most Atom users and isn't something that can or should be implemented as a community package.
- **List some other text editors or applications where this enhancement exists.**
- **Specify which version of Atom you're using.** You can get the exact version by running `atom -v` in your terminal, or by starting Atom and running the `Application: About` command from the Command Palette.
- **Specify the name and version of the OS you're using.**

Your First Code Contribution

Unsure where to begin contributing to Atom? You can start by looking through these `beginner` and `help-wanted` issues:

- Beginner issues - issues which should only require a few lines of code, and a test or two.
- Help wanted issues - issues which should be a bit more involved than `beginner` issues.

Both issue lists are sorted by total number of comments. While not perfect, number of comments is a reasonable proxy for impact a given change will have.

If you want to read about using Atom or developing packages in Atom, the Atom Flight Manual is free and available online. You can find the source to the manual in atom/flight-manual.atom.io.

Local development

Atom Core and all packages can be developed locally. For instructions on how to do this, see the following sections in the Atom Flight Manual:

- Hacking on Atom Core
- Contributing to Official Atom Packages

Pull Requests

The process described here has several goals:

- Maintain Atom's quality
- Fix problems that are important to users
- Engage the community in working toward the best possible Atom
- Enable a sustainable system for Atom's maintainers to review contributions

Please follow these steps to have your contribution considered by the maintainers:

1. Follow all instructions in the template
2. Follow the styleguides
3. After you submit your pull request, verify that all status checks are passing
   ▶ What if the status checks are failing?

While the prerequisites above must be satisfied prior to having your pull request reviewed, the reviewer(s) may ask you to complete additional design work, tests, or other changes before your pull request can be ultimately accepted.

Styleguides

Git Commit Messages

- Use the present tense ("Add feature" not "Added feature")
- Use the imperative mood ("Move cursor to..." not "Moves cursor to...")
- Limit the first line to 72 characters or less
- Reference issues and pull requests liberally after the first line
- When only changing documentation, include `[ci skip]` in the commit title
- Consider starting the commit message with an applicable emoji:
  - :art: `:art:` when improving the format/structure of the code
  - :racehorse: `:racehorse:` when improving performance
  - :non-potable_water: `:non-potable_water:` when plugging memory leaks
  - :memo: `:memo:` when writing docs
  - :penguin: `:penguin:` when fixing something on Linux
  - :apple: `:apple:` when fixing something on macOS
  - :checkered_flag: `:checkered_flag:` when fixing something on Windows
  - :bug: `:bug:` when fixing a bug
  - :fire: `:fire:` when removing code or files
  - :green_heart: `:green_heart:` when fixing the CI build
  - :white_check_mark: `:white_check_mark:` when adding tests
  - :lock: `:lock:` when dealing with security
  - :arrow_up: `:arrow_up:` when upgrading dependencies
  - :arrow_down: `:arrow_down:` when downgrading dependencies
  - :shirt: `:shirt:` when removing linter warnings

JavaScript Styleguide

All JavaScript code is linted with Prettier.

- Prefer the object spread operator (`{...anotherObj}`) to `Object.assign()`
- Inline `export` s with expressions whenever possible

```
// Use this:
export default class ClassName {

}

// Instead of:
class ClassName {

}
export default ClassName
```

- Place requires in the following order:
  - Built in Node Modules (such as `path`)
  - Built in Atom and Electron Modules (such as `atom` , `remote`)
  - Local Modules (using relative paths)
- Place class properties in the following order:

    - Class methods and properties (methods starting with `static`)
    - Instance methods and properties
- Avoid platform-dependent code

CoffeeScript Styleguide

- Set parameter defaults without spaces around the equal sign
    - `clear = (count=1) ->` instead of `clear = (count = 1) ->`
- Use spaces around operators
    - `count + 1` instead of `count+1`
- Use spaces after commas (unless separated by newlines)
- Use parentheses if it improves code clarity.
- Prefer alphabetic keywords to symbolic keywords:
    - `a is b` instead of `a == b`
- Avoid spaces inside the curly-braces of hash literals:
    - `{a: 1, b: 2}` instead of `{ a: 1, b: 2 }`
- Include a single line of whitespace between methods.
- Capitalize initialisms and acronyms in names, except for the first word, which should be lower-case:
    - `getURI` instead of `getUri`
    - `uriToOpen` instead of `URIToOpen`
- Use `slice()` to copy an array
- Add an explicit `return` when your function ends with a `for` / `while` loop and you don't want it to return a collected array.
- Use `this` instead of a standalone `@`
    - `return this` instead of `return @`
- Place requires in the following order:
    - Built in Node Modules (such as `path`)
    - Built in Atom and Electron Modules (such as `atom`, `remote`)
    - Local Modules (using relative paths)
- Place class properties in the following order:
    - Class methods and properties (methods starting with a `@`)
    - Instance methods and properties
- Avoid platform-dependent code

Specs Styleguide

- Include thoughtfully-worded, well-structured Jasmine specs in the `./spec` folder.
- Treat `describe` as a noun or situation.
- Treat `it` as a statement about state or how an operation changes state.

Example

```
describe 'a dog', ->
 it 'barks', ->
 # spec here
 describe 'when the dog is happy', ->
  it 'wags its tail', ->
  # spec here
```

Documentation Styleguide

- Use AtomDoc.
- Use Markdown.
- Reference methods and classes in markdown with the custom `{}` notation:
  - Reference classes with `{ClassName}`
  - Reference instance methods with `{ClassName::methodName}`
  - Reference class methods with `{ClassName.methodName}`

Example

```
# Public: Disable the package with the given name.
#
# * `name`    The {String} name of the package to disable.
# * `options` (optional) The {Object} with disable options (default: {}):
#   * `trackTime`     A {Boolean}, `true` to track the amount of time taken.
#   * `ignoreErrors`  A {Boolean}, `true` to catch and ignore errors thrown.
# * `callback` The {Function} to call after the package has been disabled.
#
# Returns `undefined`.
disablePackage: (name, options, callback) ->
```

Additional Notes

Issue and Pull Request Labels

This section lists the labels we use to help us track and manage issues and pull requests. Most labels are used across all Atom repositories, but some are specific to `atom/atom`.

GitHub search makes it easy to use labels for finding groups of issues or pull requests you're interested in. For example, you might be interested in open issues across `atom/atom` and all Atom-owned packages which are labeled as bugs, but still need to be reliably reproduced or perhaps open pull requests in `atom/atom` which haven't been reviewed yet. To help you find issues and pull requests, each label is listed with search links for finding open items with that label in `atom/atom` only and also across all Atom repositories. We encourage you to read about other search filters which will help you write more focused queries.

The labels are loosely grouped by their purpose, but it's not required that every issue has a label from every group or that an issue can't have more than one label from the same group.

Please open an issue on `atom/atom` if you have suggestions for new labels, and if you notice some labels are missing on some repositories, then please open an issue on that repository.

Type of Issue and Issue State

Label name	`atom/atom` :mag_right:	`atom-org` :mag_right:	Description
`enhancement`	search	search	Feature requests.
`bug`	search	search	Confirmed bugs or reports that are very likely to be bugs.
`question`	search	search	Questions more than bug reports or feature requests (e.g.

			how do I do X).
`feedback`	search	search	General feedback more than bug reports or feature requests.
`help-wanted`	search	search	The Atom core team would appreciate help from the community in resolving these issues.
`beginner`	search	search	Less complex issues which would be good first issues to work on for users who want to contribute to Atom.
`more-information-needed`	search	search	More information needs to be collected about these problems or feature requests (e.g. steps to reproduce).
`needs-reproduction`	search	search	Likely bugs, but haven't been reliably reproduced.
`blocked`	search	search	Issues blocked on other issues.
`duplicate`	search	search	Issues which are duplicates of other issues, i.e. they have been reported before.
`wontfix`	search	search	The Atom core team has decided not to fix these issues for now, either because they're working as intended or for some other reason.
`invalid`	search	search	Issues which aren't valid (e.g. user errors).
`package-idea`	search	search	Feature request which might be good candidates for new packages, instead of extending Atom or core Atom packages.
`wrong-repo`	search	search	Issues reported on the wrong repository (e.g. a bug related to the Settings View package was reported on Atom core).

Topic Categories

Label name	`atom/atom` :mag_right:	`atom-org` :mag_right:	Description
`windows`	search	search	Related to Atom running on Windows.
`linux`	search	search	Related to Atom running on Linux.
`mac`	search	search	Related to Atom running on macOS.
`documentation`	search	search	Related to any type of documentation (e.g. API documentation and the flight manual).
`performance`	search	search	Related to performance.
`security`	search	search	Related to security.
`ui`	search	search	Related to visual design.
`api`	search	search	Related to Atom's public APIs.
`uncaught-exception`	search	search	Issues about uncaught exceptions, normally created from the Notifications package.

`crash`	search	search	Reports of Atom completely crashing.
`auto-indent`	search	search	Related to auto-indenting text.
`encoding`	search	search	Related to character encoding.
`network`	search	search	Related to network problems or working with remote files (e.g. on network drives).
`git`	search	search	Related to Git functionality (e.g. problems with gitignore files or with showing the correct file status).

`atom/atom` Topic Categories

Label name	`atom/atom` :mag_right:	`atom-org` :mag_right:	Description
`editor-rendering`	search	search	Related to language-independent aspects of rendering text (e.g. scrolling, soft wrap, and font rendering).
`build-error`	search	search	Related to problems with building Atom from source.
`error-from-pathwatcher`	search	search	Related to errors thrown by the pathwatcher library.
`error-from-save`	search	search	Related to errors thrown when saving files.
`error-from-open`	search	search	Related to errors thrown when opening files.
`installer`	search	search	Related to the Atom installers for different OSes.
`auto-updater`	search	search	Related to the auto-updater for different OSes.
`deprecation-help`	search	search	Issues for helping package authors remove usage of deprecated APIs in packages.
`electron`	search	search	Issues that require changes to Electron to fix or implement.

Pull Request Labels

Label name	`atom/atom` :mag_right:	`atom-org` :mag_right:	Description
`work-in-progress`	search	search	Pull requests which are still being worked on, more changes will follow.
`needs-review`	search	search	Pull requests which need code review, and approval from maintainers or Atom core team.
`under-review`	search	search	Pull requests being reviewed by maintainers or Atom core team.
`requires-`	search	search	Pull requests which need to be updated based on review comments and then reviewed again.

changes			
needs-testing	search	search	Pull requests which need manual testing.