Contributing to Hazel!

======================

Thanks for your interest in contributing to Hazel! :tada: We love getting [pull requests](https://www.quora.com/GitHub-What-is-a-pull-request) for bugfixes and contributions of our community to keep Hazel growing.

We want to keep it as easy as possible to contribute changes. These guidelines are intended to help smooth that process, and allow us to review and approve your changes quickly and easily. Improvements are always welcome! Feel free to [open an issue][issue-tracker] or [submit a new pull request][submit-pr]. And finally, these are just guidelines, not rules, so use your best judgement when necessary.

If you're new to [GitHub][github], you may want to begin with [Getting Started with GitHub](https://help.github.com/en/categories/getting-started-with-github) and Thinkful's [GitHub Pull Request Tutorial](https://www.thinkful.com/learn/github-pull-request-tutorial/).

## Language

As TheCherno creates his videos about Hazel in English, the devs ask to keep the code base, issues and pull requests in English only.

Thanks for your understanding.

## Reporting Bugs

Bugs should be reported on our [GitHub Issue Tracker][issue-tracker] using the bug report template.

Follow the advice in [How do I ask a good question?][how-to-ask]. While the article is intended for people asking questions on Stack Overflow, it all applies to writing a good bug report too.

## Requesting New Features

Feature requests should also be sent to our [GitHub Issue Tracker][issue-tracker] using the feature request template.

- Explain the problem that you're having, and anything you've tried to solve it using the currently available features.

- Explain how this new feature will help.

- If possible, provide an example, like a code snippet, showing what your new feature might look like in use.

Also, much of the advice in [How do I ask a good question?][how-to-ask] applies here too.

## Contributing a Fix or Feature

You've created a new fix or feature for Hazel. Awesome!

1. If you haven't already, create a fork of the Hazel repository.

2. Create a topic branch, and make all of your changes on that branch.

3. Submit a pull request, use the implemented issue template if it is based on an issue or the new issue template if it is not linked to any issue.

4. Give us a moment. Hazel is maintained by only a few people, all of whom are doing this on their limited free time, so it may take us a bit to review your request. Bug fixes should be merged in directly, while features usually require Cherno's approval with or without it mentioned in one (or more) videos.

If you're not sure what any of that means, check out Thinkful's [GitHub Pull Request Tutorial][thinkful-pr-tutorial] for a complete walkthrough of the process.

### Writing a Good Pull Request

- Stay focused on a single fix or feature. If you submit multiple changes in a single request, we may like some but spot issues with others. When that happens, we have to reject the whole thing. If you submit each change in its own request it is easier for us to review and approve.

- Limit your changes to only what is required to implement the fix or feature. In particular, avoid style or formatting tools that may modify the formatting of other areas of the code.

- Use descriptive commit titles/messages. "Implemented \<feature\>" or "Fixed \<problem\> is better than "Updated \<file\>".

- Make sure the code you submit compiles and runs without issues. When we set up unit tests and continuous integration we also expect that the pull request should pass all tests.

- Use [closing keywords][github-help-closing-keywords] in the appropriate section of our Pull Request template where applicable.

- Follow our coding conventions, which we've intentionally kept quite minimal.

### Coding Conventions

- Naming convention:

- For functions we use pascal case: \*\*`FunctionName`\*\*.

- For (scoped) variables and function parameters we use camel case: \*\*`variableName`\*\* and \*\*`parameterName`\*\*.

- For class names we use pascal case: \*\*`ClassName`\*\*.

- For class variables we use the Hungarian notation:

- Class member variables get the 'm\_' prefix: \*\*`m\_ClassMemberVariableName`\*\*.

- Class static variables get the 's\_' prefix: \*\*`s\_ClassStaticVariableName`\*\*.

- For macros we use snake case: \*\*`MACRO\_NAME`\*\*.

- If it is specifically related to Hazel, we add the 'HZ\_' prefix: \*\*`HZ\_MACRO\_NAME`\*\*.

- If there is a macro for the application and for the engine, we add an additional 'CORE\_' prefix to the engine macro: \*\*`HZ\_CORE\_MACRO\_NAME`\*\*.

- Use tabs for indentation, not spaces.

- When in doubt, match the code that's already there.

[github]: https://github.com

[how-to-ask]: https://stackoverflow.com/help/how-to-ask

[issue-tracker]: https://github.com/TheCherno/Hazel/issues

[submit-pr]: https://github.com/TheCherno/Hazel/pulls

[thinkful-pr-tutorial]: https://www.thinkful.com/learn/github-pull-request-tutorial/

[github-help-closing-keywords]: https://help.github.com/en/articles/closing-issues-using-keywords