Our GitHub community is one of the biggest reasons of our success. Every single day, you create between 50 - 250 issues with inspiring ideas for new functionality, you tell us what works as expected and what doesn't, and you tell us how to improve what's already there. We appreciate every single issue you create. Thank you!

Despite our best efforts to deal with all of those issues, over the course of time we fall behind. When you look at an issue it's sometimes unclear whether or not we intent to provide the functionality you propose, whether or not we fix the bug, whether or not the issue is a duplicate of another one.

In order to restore a reasonable level of clarity we annually perform a House Keeping Iteration in which we go through all open issues. We categorize, label, fix, and close issues. This results in a wave of notification which is hard to deal with (our apologies) but in the end there is better understanding of what will happen to your ideas.

Below are more details about what we do during a House Keeping Iteration.

Closing Issues

We close issues for the following reasons:

Reason	Label
The issue is obsolete or already fixed.	
We didn't get the information we need within 7 days.	`needs more info`
It's a duplicate of another issue.	`*duplicate`
The feature request is best implemented by an extension.	`*extension-candidate`
What is described is the designed behavior.	`*as-designed`
The issue was caused by an extension.	`*caused-by-extension`
The issue is a developer question.	`*dev-question`
The issue is a user question.	`*question`
The issue is off-topic, does not contain any actionable information or was unintended.	`invalid`
Given the information we have we can't reproduce the issue.	`*not-reproducible`
The feature request is out of scope. (See below)	`*out-of-scope`

We close issues with the help of a bot that responds to a particular comment such as `/duplicate of #1234` or to assigning a label with adding a pre-canned comment to the issue and closing the issue.

Out-of-Scope Feature Requests

Feature requests like all issues are a means of communication between us and our community as well as among members of the community. Thus, in principle, we could keep all feature requests open no matter what will happen to the feature they describe. But this makes it hard for you to understand what has realistic chances to ever make it into the repository. We therefore close feature requests we won't address.

If you are the author of a feature request you might not like that we close your issue. It might even feel like a slap in your face. We understand that. All of us have been there as well - in this project or others we have contributed to. So, be assured, we love all of your input. Don't take personal offense when we decide to close your issue :peace_symbol:. If you feel your feature request deserves to stay open, improve your use case or gather more up-votes. Then ping us and we'll consider reopening the feature request.

Here are the criteria we use to make the decision about closing a feature request:

1. Does the functionality described in the feature request have any reasonable chance to be implemented in the next 24 months? 24 months is longer than our roadmap which outlines the next 6-12 months. Thus, there is some crystal ball reading on our part, and we'll most likely keep more feature requests open than what we can accomplish in 24 months.
2. Has the community at large expressed interest in this functionality? I.e. has it gathered more than 10 up-votes or more than 10 comments? This criterion alone covers more than 650 of the 2850 open feature requests as of right now, October 9th, 2019.
3. Do we think the feature request is bold and forward looking and would we like to see it be tackled at some point even if it's further out than 24 months? (Clearly, this one is the most subjective criterion.)

If the answer to any of the three questions is `yes` then we ask about affordability:

4. Can our team afford to implement the feature? I.e. are the costs to implement the functionality reasonable compared to the size of our team?

In summary, a feature needs to pass one of 1-3 and 4. Otherwise, we'll close it as `out of scope`. We'll unsubscribe from these issues to reduce the overall number of issue notifications we receive every day.

Won't fix Bugs

We close bugs as `won't fix` if there is a negative cost-benefit balance. It's not that we don't care about users who are affected by an issue but, for example, if the fix is so complex that despite all of our tests we risk regressions for many users, fixing is not a reasonable choice. When we close a bug as `won't fix` we'll make our case why we do so.

Upstream Issue

We label some issues as `upstream`. Upstream means that the issue is in a package or library that we consume and that we cannot fix independently. We close an `upstream` issue if we can establish a clear traceability link between the issue in our repository and an issue in the issue tracker of the package or library. In some cases this is not possible, for example, we might have identified an issue as a Chromium issue but Chromium does not accept the issue yet because the repro case is too complex.

Categorizing Issues

Each issue must have a **type** label. Most type labels are grey, some are yellow. Bugs are grey with a touch of red.

Type	Description
`bug`	the implementation of a feature is not correct
`feature-request`	request for a new feature
`under-discussion`	not decided whether the issue is a bug or feature
`debt`	improve the implementation/architecture

`needs more info`	not possible to assign a type label due to missing information
`question`	we should direct questions to SO
`upstream`	an issue used to track an issue in an upstream component
`electron`	an issue with electron
`engineering`	issues related to our engineering system or our processes

Labeling Issues

- We assign the `important` label to issues that
  - result in data loss
  - a breakage of extension
  - critical security, performance issues
  - UI issue that makes a feature unusable
- We label issues that the community can take up as `help-wanted`.
- If issues are suitable for beginners we may add the `good-first-issue` label and we add code pointers that help beginners to get started with a PR.

Fixing Issues

- We assign bugs that we plan to fix during the grooming to the milestone of the current iteration.