tl;dr

- Assign bugs to yourself if you're working on them (ignore this if you don't yet have the permission to assign yourself).
- Unassign bugs you are not working on soon.
- If an issue is not assigned, assume it is available to be worked on.
- Use thumbs-up on the issue description to help the team prioritise.
- P0-P2 bugs are critical and get examined for progress every week. P3 is the highest priority a bug can normally have.

Overview

We use three issue trackers: the main one on flutter/flutter, one for the flutter.dev Website, on flutter/website, and one for the IntelliJ and Android Studio plugins, on flutter/flutter-intellij.

This page mostly talks about how we handle things for the flutter/flutter issue tracker.

Issue philosophy

We assume that Flutter, like all non-trivial software, has an infinite number of bugs. The issue tracker contains the list of bugs that we are very lucky to have had reported by our generous community.

Within the bug database we try to make sure each issue is actionable and discoverable. We do this by carefully updating the issue subject line, making sure every issue has steps to reproduce, and using labels to categorize the issue in ways that can be found by GitHub search.

Labels

We use many labels.

The `severe:` prefix indicates labels regarding a level of severity (e.g. regressions, new features, crashes). Severity in and of itself does not say how fast we will fix the bug; rather it provides clues as to the nature of the defect, just as other labels do. A bug may have the `severe: crash` label, but e.g. if it relates to the `flutter` tool crashing when run with the system date set to 1998, we are not likely to consider it a high priority.

Priorities

See also: [[What should I work on?]]

The `P0` label indicates a critically dire issue such as a build break, regression, or failure in existing features that keeps us from shipping the current build. We look at issues with this label frequently. If you find yourself assigning a `P0` label to an issue, please be sure that there's a positive handoff between filing and a

prospective owner for the issue. P0 bugs should be rare. Normally there should be zero open P0 bugs. "The world is on fire."

The `P1` label indicates that the issue deserves immediate attention, such as one actively blocking a top-tier customer who is trying to ship. (See below under "customers" for a definition of "top-tier customer".) There are generally only one or two such P1 bugs at a time, and we aim for there to be none. "A customer is on fire." We also use this as the initial priority for critical infrastructure problems like flaky tests. We aim for zero of these also but currently we are working through some technical debt that's we've accumulated so there are a few more.

The `P2` label indicates that the issue is about to block a top-tier customer or is an important item of technical debt that we want to fix as soon as possible. There are generally less than twenty-five P2 bugs (one GitHub search results page). These may not be worked on immediately (sometimes some P3 issues are handled first, e.g. when they affect a lot of people, or are particularly egregious), but are top-of-mind and we explicitly consider them weekly. "A customer is about to be on fire."

The `P3` label indicates high-priority issues that are at the top of the work list. This is the highest priority level a bug can have if it isn't affecting a top-tier customer or breaking the build. Bugs marked P3 are generally actively being worked on unless the assignee is dealing with a P0-P2 bug (or another P3 bug).

The `P4` label indicates issues that we agree are important to work on, but not at the top of the work list. This is the default level for bugs. A bug at this priority level may not be fixed for a long time. Often a bug at this level will first migrate to P3 before we work on them.

The `P5` label indicates issues we think are valid but not important. This is the default level for new feature requests. We are unlikely to work on such issues unless they form part of a long-term strategic plan (see the [[Roadmap]]).

The `P6` label indicates valid issues that are unlikely to ever be worked on. We use "thumbs-up" on these issues to promote them to `P5` or higher based on demand.

During normal work weeks (e.g. not around the new year), issues marked P0-P2 get audited weekly during the "critical triage" meeting to ensure we do not forget about them. Issues marked P0 should get updates multiple times a day, and issues marked P1 should get updates multiple times a week, to keep the rest of the team (and anyone affected by the issues) appraised of progress. Updates might be less frequent when a fix is blocked on some other issue and thus progress cannot be made, but even then, updates on P0 and P1 issues should not be less frequent than weekly.

See also: [[Triage]], which describes how we go through bugs and make sure they are sorted accordingly.

Adding labels

Labels are more or less free, so we can add them pretty easily. Please mention it to other team members first, so that they know what you are planning and can give feedback. Please make sure labels use a consistent color and naming scheme (e.g. all the framework-related labels are blue and start with `f:`).

Labels should be used for adding information to a bug. If you plan to use a label to find all instances of a particular topic (e.g. finding all PRs where someone wrote a design doc), be aware that there's no way to force people to label issues or PRs. You can, however, rely on automation to do it, for example you could write a script that labels all PRs that affect the framework.

Customers

The Flutter team is formed of engineers from many sources, including dedicated volunteers and employees of companies like Google. Each of these may have different ideas of who their customers are. For example, Google engineers consider some Google teams to be their customers, but someone who contributes on a code-for-hire basis may have their own customers.

Some teams using Flutter have a special relationship with some members of the Flutter team (e.g. they're collaborating with us on a new feature, or they're working with us on a product demo for an upcoming event). This is usually a fairly short-term arrangement for a specific business purpose. We provide such customers with a label (`customer: ...`) in our GitHub issue tracker. When these customers are working closely with members of the Flutter team, we may consider them "top-tier customers" for the purposes of prioritization.

Priorities `P1` and `P2` (see above) are reserved for bugs that affect these top-tier customers.

Coordinating between bug systems Some customers have their own separate bug systems, in which they track Flutter issues. We consider our GitHub issue list to be canonical. However, if there is a link from the issue in our bug system to the customer's issue in their bug system, and we have been granted access, we will follow that link and may communicate in that separate bug system when attempting to track down the issue.

Special customer labels The `customer: product` label is used to bring issues that product management and senior leads want resolved to the attention of the appropriate engineering team.

The `customer: crowd` label is used to represent bugs that are affecting large numbers of people; during initial [[Triage]], high-profile bugs get labeled in this way to bring them to the attention of the engineering team. "Large numbers" is a judgement call. If dozens of people independently run into the same issue and file a bug and we end up realizing that they're all duplicates of each other,

then that would be a good candidate. On the other hand, if there is an active campaign to get people to comment on a bug, then it's probably not legitimately a `customer: crowd` bug, because people generally report bugs without having to be convinced to do so.

In general, a bug should only be marked `customer: crowd P1` if it is so bad that it is literally causing large numbers of people to consider changing careers. Similarly, a bug should only be marked `customer: crowd P2` if people are threatening to never upgrade to the next release because of this bug.

Thumbs-up reactions

To vote on an issue, use the "Thumbs-up" emoji to react to the issue.

When examining issues, we use the number of thumbs-up reactions to an issue as a guide to its priority.

See also:

- All open issues sorted by thumbs-up
- Feature requests by thumbs-up
- Bugs by thumbs-up

We ignore other emoji reactions.

Milestones

We do not use GitHub milestones to track work.

Assigning Issues

Issues are typically self-assigned. Only assign a bug to someone else if they have explicitly volunteered to do the task. If you don't have permissions to assign yourself an issue you want to work on, don't worry about it, just submit the PR (see [[Tree Hygiene]]).

Only assign a bug to yourself when you are actively working on it or scheduled to work on it. If you don't know when you'll be working on it, leave it unassigned. Similarly, don't assign bugs to people unless you know they are going to work on it. If you find yourself with bugs assigned that you have not scheduled specific time to work on, unassign the bug so that other people feel empowered to work on them.

Do assign a bug to yourself if you are working on it, or if you have scheduled time to work on them and are confident you will do so! This is how people can figure out what is happening. It also prevents duplicate work where two people try to fix the same issue at once.

You may hear team members refer to "licking the cookie". Assigning a bug to yourself, or otherwise indicating that you will work on it, tells others on the team to not fix it. If you then don't work on it, you are acting like someone who has taken a cookie, licked it to be unappetizing to other people, and then not eaten it. By extension, "unlicking the cookie" means indicating to the rest of the team that you are not actually going to work on the bug after all, e.g. by unassigning the bug from yourself.

File bugs for everything

File bugs for anything that you come across that needs doing. When you implement something but know it's not complete, file bugs for what you haven't done. That way, we can keep track of what still needs doing.

Exceptions

Do *not* file bugs that meet the following criteria:

- Asking meta-questions like "why was bug #XYZ closed?" Instead, post on the original issue or raise the actual problem that is still not resolved.
- Intentional duplicates like "This is the same as bug #ABC but that one is not getting enough attention." Instead, upvote the original issue or add a comment that provides new details that are not already captured or (best of all) assign it to yourself and start working on it!

How to propose a specific change

If you have an idea that you would like to land, the recommended process is:

1. File a bug describing the problem.
2. Write a design doc that references this problem and describes your solution.
3. Socialize your design on the bug you filed and on [[Chat]]. Collect feedback from various people.
4. Once you have received feedback, if it is mostly positive, implement your idea and submit it. See the [[Tree Hygiene]] wiki page for details on submitting PRs.

Every issue should be actionable

Avoid filing issues that are on vague topics without a clear problem description.

Please close issues that are not actionable.

Issues should have clear steps to reproduce Every issue should have a clear description of the steps to reproduce the problem, the expected results, and the actual results.

If an issue is lacking this information, request it from the commenter and close the issue if information is not forthcoming.

Comments

Consider posting issues in English

If you are able to read and write English clearly, consider posting your issue in English, even if it is about a language specific issue (like the way text renders in some non-English language).

It is fine to post issues in languages other than English, but consider that many readers will rely on automatic translation services to read your issue. Please avoid using screenshots in languages other than English, as services like Google Translate will not translate the text in images, and the pool of people able to assist you will be reduced.

Do not add "me too" or "same" comments to bugs

The Flutter team prioritizes issues in part based on the number of +1 (thumbs up) reactions on the top level comment of the bug. Adding comments like "me too" or "same here" is generally distracting and makes it harder to find other more meaningful content in the bug. If you have no new details to add, consider just thumbs up-ing the issue. If you wish to subscribe to the issue, click the "subscribe" button in the right hand column of the GitHub UI.

Adding comments explaining how a bug is dire and how you will stop using Flutter if it is not fixed is upsetting for the engineers working on Flutter (many of whom are volunteers, not that being paid to work on Flutter makes such comments any less upsetting). Out of a respect for the team, and as required by our code of conduct, we ask that you avoid adding comments that are not actively helpful. There are other venues if you want to complain without being constructive.

Issues are not always the best venue for discussions

Discussions within an issue should remain focused on the topic, specifically about what the filed issue is and how to solve it. Broader discussions are best suited to happen on Discord or in design docs using Google Docs (see [[Chat]]). This is because GitHub hides comments, doesn't have threading, notifications get lost in the swamp of other GitHub e-mails, etc.

If you move to another tool for part of the discussion, remember to add a summary of the discussion and document any decisions that took place. This allows people following the issue to keep updated and continue to participate.

Issues are never an appropriate venue for asking for help with your code. Issues are also not a good venue for discussing project direction.

Comments providing workarounds

Providing workarounds for issues can be helpful for developers using Flutter and finding a bug, but please keep such comments to a minimum so as to

avoid disrupting the engineers trying to fix the issue. Rather than discussing workarounds, provide a pointer to another forum (e.g. Stack Overflow) where workarounds and temporary solutions are more appropriate. Thanks.

Avoid posting screenshots of text

If you want to show code, quote someone, or show a string of text that does not render properly with Flutter, please avoid sharing it via an image or screenshot. Text in images cannot be copied, and cannot be automatically translated via services like Google Translate. This makes it harder for team members who do not speak that language to participate in the issue.

It is perfectly fine to share a screenshot of text rendering invalidly, but also include the actual string or character(s) that lead to it so that they can be copied and pasted into a test case.

Provide reduced test cases

To debug a problem, we will need to be able to reproduce it. The best way to help us do that is to provide code, licensed according to the BSD license used by Flutter, that has been reduced as far as possible (such that removing anything further stops showing the bug). Attach such a file or files to the issue itself.

For legal reasons, we cannot debug problems that require looking at proprietary code or, generally, code that is not publicly available.

When will my bug be fixed?

To determine when a bug will be fixed, look at the priority and milestone.

If it has a milestone, that is roughly when we hope to have it fixed by.

If it has no priority, it may not yet have been triaged.

If the priority is labeled as a `P0` to `P3`, we are likely to address in the near term; we just need to find time.

If the issue is labeled as a `P4`, we are not planning to work on it anytime soon, but we agree it's something we should work on in the coming years.

If the issue is labeled `P5` or `P6`, we are not currently expecting to ever work on it (especially `P6`).

See also [[Popular issues]].

Escalating an issue that has the wrong priority

If you have a relationship with the Flutter team, raise the issue with your contact if you think the priority should be changed.

If you don't, consider finding like-minded developers to either implement the feature as a team, or to fund hiring someone to work on the feature, or to mark the issue with a thumbs-up reaction. We regularly look through all issues sorted by the number of thumbs-up reactions to determine the relative priority of upcoming work, so that is one way to change our mind.

Please don't comment on an issue to indicate your interest. Comments should be reserved for making progress on the issue.

Locking an issue

Closed issues that haven't received any activity in a few weeks are automatically locked by a bot. This is done to encourage developers to file new bugs, instead of piling comments on old ones.

Under normal circumstances, open issues should not regularly be locked. The most common reason for manually locking an open issue is that issue is well understood by the engineers working on it, is believed to be appropriately prioritized, has has a clear path to being fixed, and is otherwise attracting a lot of off-topic or distracting comments like "me too" or "when will this be fixed" or "I have a similar issue that might or might not be the same as this one".

If you are concerned that such an issue is not receiving its due attention, see Escalating an Issue, described above. If you are not already a contributor but would like to work on that issue, consider reaching out on an appropriate chat.

If you have a similar issue and are not sure if it is the same, it is fine to file a new issue and linking it to the other issue. Please avoid intentionally filing duplicates.

Very rarely, an issue gets locked because discussion has become unproductive and has repeatedly violated the Code of Conduct.

Flaky tests

When a test flakes, a P1 bug is automatically filed with the label `team: flakes`. This issue should be investigated with all due haste, and a priority level should then be assigned to the issue. At any particular time, the most flaky tests should remain P1. However, flakes that are hard to pin down may be downgraded in priority (e.g. to P3 or P4). Please do not ignore the issue entirely, however, and make sure to close bugs once they are resolved, even if it's by magic.

See also: [[Reducing test flakiness]]