The Flutter project has many teams:

- Design languages, covering:
  - The material library (flutter/flutter packages/flutter/lib/src/material, "f: material design")
  - The cupertino library (flutter/flutter packages/flutter/lib/src/cupertino, "f: cupertino")
- The Flutter command line tool (flutter/flutter packages/flutter_tools, "tool")
- Ecosystem (flutter/plugins, flutter/packages and the plugin infrastructure in flutter/flutter, "plugins", "packages")
- The Flutter framework (everything else in flutter/flutter, "framework")
- The Engine (flutter/engine and flutter/buildroot, "engine")
- Mobile Platforms team (some code in flutter/flutter and flutter/engine), "platform-android", "platform-ios".
- Web (some code in flutter/flutter and flutter/engine, "platform-web")
- Desktop (some code in flutter/flutter and flutter/engine, "a: desktop")
- Developer experience (e.g. the devtools package)
- User Experience Research
- Developer Relations
- Infrastructure (mainly flutter/cocoon and flutter/flutter dev/devicelab, "team: infra")

There are also specific cross-cutting areas of work that affect multiple subteams (e.g. accessibility, add-to-app, dynamic code loading, etc).

We also work closely with other projects, such as Dart and Skia, and with many customers.

Responsibilities

Subteams are responsible for reviewing PRs in their area, triaging issues, and scheduling work. How subteams organize themselves is not defined by this document. This document does not attempt to impose a process, merely a set of responsibilities.

Reviewing PRs

Please review PRs in your area (based on label and/or repositories). The goal is to have a prompt turnaround for all PRs.

Please have an OKR around handling of PRs with the relevant label and/or in the relevant repository.

If a contributor does not have time to deal with a PR's comments (e.g. they don't want to add tests), please take over that PR if it seems valuable, or close it if not (encouraging them to file a new PR once they have the time to address comments).

Please don't leave lingering stale PRs open. All PRs should be actively being worked on.

Issues and scheduling

For 2019, please focus on: 1. Testing (coverage, flakiness; so that we can cut builds at almost any time with great confidence that they do not have regressions) 1. Quality (fixing bugs, prioritised by thumbs-up and customer requests; so that developers and end users run into fewer bugs) 1. Documentation (coverage and quality, diagrams, and sample code in documentation; so that developers can be more productive when using Flutter).

See our roadmap for more details: https://github.com/flutter/flutter/wiki/Roadmap

Please triage issues with your label on a regular basis. You may do this in whatever manner you prefer (on your phone while in line for lunch, as a team exercise in a dedicated meeting room, by having some sort of team rotation, whatever).

You must cover these bug lists in particular: https://github.com/flutter/flutter/wiki/Triage#area-focused-triage

- Assign bugs that you are working on or that you have committed to work on.
- Unassign bugs you are not working on and have no specific scheduled plans to work on.
- Make sure that assigned bugs have a month-based milestone (see section below).

See our page on managing issues: https://github.com/flutter/flutter/wiki/Issue-hygiene

Keep an eye out for bugs that should block releases, update the bad builds page accordingly: https://github.com/flutter/flutter/wiki/Bad-Builds

Please send a representative to the weekly critical triage meeting on Tuesdays. (Currently this is only possible for people at Google, but we're working on making this open. Please stay tuned.)

Scheduling

Put bugs in the appropriate dated milestones once you have scheduled the work. Bugs in a milestone should be assigned to the engineer who has committed to do the work, and they should be confident that they will do the work by the given milestone.

As a team, please let @Hixie know the timelines for big new features that you have planned (via whatever channel you like), so that we can work with the product organization to announce them accordingly, and so that he can update the roadmap accordingly.

Be conservative when scheduling. Customers always assume things will be done sooner than you promise, and there are always going to be emergencies, so you need slack in your schedule.

You will be more effective, more popular, and your morale will be higher, if you focus on a small set of things and really knock those out of the park, than if you try to do a large number of things and only do a little bit for each, so aim to underpromise and overdeliver.

This may mean your OKRs are more optimistic than what you report as your scheduled timeline. With OKRs we generally try to hit 67% of the plan. With the roadmap we want to hit 150%.

(OKRs are how some team members plan their work, notably it is used by Google engineers.)

It also means milestones should be conservative and should never slip (it's fine to be early, it's bad to be late, because our customers are depending on us to deliver by the date we said we would deliver by).