Flutter's core values

🏗 Build the best way to develop user interfaces.

This tells us what we are creating. It helps narrow our focus.

The best way to develop user interfaces is a *productive* way of developing.

The best way to develop user interfaces creates *beautiful* applications.

The best way to develop user interfaces creates *fast* applications.

The best way to develop user interfaces enables developers to create applications *fast*.

The best way to develop user interfaces is *extensible*, so that we are not a barrier to developers.

🔎 Focus on the user and all else will follow.

Our user is the developer. Our developer's user is the end user of the application written with Flutter.

Our first priority is to our developer's user, the end user. Our second priority is to our user, the developer.

Caring about the end user means having high quality support for accessibility, top performance, stability, high fidelity and compatibility with the user's platform, supporting low-end devices, and so forth.

Caring about the developer means creating a joyful and productive development experience with quick iteration cycles, creating usable, simple, reliable, predictable APIs, giving the developer full access to the underlying platform, and so forth.

We respect our users, whoever they are.

📖 Openness

Flutter is an open source project, in the full senses of the word: we are open to ideas, we are open to contributions, our code and our roadmap are open, our priorities are open. Transparency leads to a higher quality product. While the most active part of our team is currently primarily formed of employees from Google, we are growing and welcome anyone to join the team; at this point, the majority of people with commit access are not part of Google's Flutter team. Flutter's stewardship is managed by a team at Google.

🚤 Maintaining quality

A mediocre product cannot be the best way to develop user interfaces, so we must build processes around maintaining high levels of quality.

This manifests in various ways. One is that we are feature-driven, not date-driven: we do not plan work based on deadlines. We may sometimes host events where we announce new features, but these events will announce features that have become available, rather than the features becoming available in order to be announced. This means sometimes a feature we intended to announce will slip and not be announced, but we prefer this to announcing a rushed feature.

We may sometimes gate features behind flags until we are confident of their quality.

🤣 Have fun doing it

Last, but definitely not least, we want to make sure that our work environment is pleasant for everyone involved. Your health and the health of your family and friends is more important than Flutter. Our community is welcoming. We don't know everything; all of us can make mistakes.

Support

When we think about whether we claim to "support" something, e.g. whether Flutter supports Windows 7, we use the following frame of reference. We document which platforms we consider to fall into each category on flutter.dev.

Areas of support

We distinguish several categories along which one can describe a level of support:

- Supporting for deployment (we support a platform for deployment if you can run release builds of applications on that platform).
- Supporting for *development* (we support a platform for development if you can run debug builds of applications on that platform, connect them to a debugger, hot reload, view logs, and so forth; we support a plugin for development if it can be stepped through in a debugger).
- Supporting for *hosting* development (we support a platform for hosting development if it is one on which you can compile a Flutter application; we support an IDE for hosting development if it has a Flutter plugin).
- Supporting for Flutter development (we support a platform for Flutter development if one can contribute to Flutter itself from that platform).

The list of supported platforms on flutter.dev is describing the platforms supported for deployment.

Levels of support

For each area, we consider the level to which we provide support:

1. We will literally help you with your code if things don't work. This is very rare. (See also "top-tier customers".)
2. We will make a best effort to ensure that well written code works (e.g. we have testing on that platform). This is a common level for target platforms that have reached a label of "stable" (e.g. Android, iOS) on devices that are widely available (e.g. 64bit ARM). This corresponds to the "Supported Google-tested platforms" category on the list of supported platforms.
3. We will not go out of our way to prevent code from working, but if it works it's because of luck and our best intentions rather than a coordinated effort (e.g. we do no testing on that platform). This is a common level of support for less commonly-used devices. For example, we do minimal testing of 32bit iOS devices. This corresponds to the "Best effort platforms tested by the community" category on the list of supported platforms.
4. We will pay no attention to whether code works, but we will accept patches. This is a common level of support for the many esoteric embedded platforms that we have no way to even manually test. For example, if you have your own SoC and are running Flutter, we want you to succeed but we don't have any way to ensure it keeps working. If you are willing to provide reliable CI support for your platform, we are more than happy to work with you to move that platform to level 2.
5. We won't accept patches. This is the appropriate response for features and platforms that are wildly outside our roadmap. For example, maintaining a Rust port of the framework is not something the Flutter project would accept patches for. This corresponds to the "Unsupported platforms" category on the list of supported platforms.

See also:

- Code of Conduct
- Contributor Guide
- Flutter's Culture of Inclusivity
- Flutter culture and how to preserve it